Order-Preserving Encryption for the Confidential Inference in Random Forests: FPGA Design and Implementation

Rupesh Raj Karn
Khalifa University
Sas Al Nakhl, Abu Dhabi, UAE

Kashif Nawaz
Technology Innovation Institute
Masdar City, Abu Dhabi, UAE

Ibrahim (Abe) M. Elfadel
Khalifa University
Sas Al Nakhl, Abu Dhabi, UAE

ABSTRACT

Prior work has addressed the problem of confidential inference in decision trees. Both traditional order-preserving cryptography (OPE) and order-preserving NTRU cryptography have been used to ensure data and model privacy in decision trees. Furthermore, FPGA architectures and implementations have been proposed for implementing such confidential inference algorithms on resource-limited, edge-based platforms such as low-cost FPGA boards. In this paper, we address the challenging problem of scalability of order-preserving confidential inference to random forests, which are ensembles of decision trees that are meant to improve their classification accuracy and reduce their overfitting. The paper develops a methodology and an FPGA implementation strategy for scaling up OPE to random forests. In particular, a framework is used to study the multifaceted tradeoffs that exist between the number of trees in the random forest, the strength of the encryption, the accuracy of the inferences, and the resources of the edge platform. Extensive experiments are conducted using the MNIST dataset and the Intel DE10 Standard FPGA board.

KEYWORDS

Decision Tree, Random Forest, Order-preserving Encryption, Confidential Inference, FPGA, Sequential Circuit, Combinational Circuit

1 INTRODUCTION

The aim of privacy-preserving machine learning [1–3] is to reconcile the divergence between privacy concerns and the advantages of machine learning (ML). This paper illustrates an approach to address ML classification challenges by employing a fusion of random forest (RF) and privacy methodologies. Model privacy ensures that a malicious entity cannot compromise the model. Given that ML models often represent significant intellectual property for many companies, the risk of model theft carries substantial financial risks. On the other hand, data privacy ensures that an adversarial entity cannot reverse-engineer the training/validation data. Recent research [4] has shown that reconstructing training data and reverse engineering models are more feasible than commonly perceived. Privacy-preserving ML techniques encompass perturbation methods such as differential privacy [5], cryptographic approaches such as homomorphic encryption and multiparty computing [6], as well as ML-specific strategies like federated learning [2]. In the context of RFs, previous works [1–3, 6, 7] has used various homomorphic encryption techniques to protect privacy, during both the training and inference phases. In this paper, we use OPE, a cryptographic algorithm that is well suited for RFs [8, 9]. We focus on ensuring the privacy of the inference phase on an edge platform where the confidentiality of the provider's model and user's data is most challenging to achieve.

In [10], it is demonstrated that preserving the confidentiality of decision tree inference is attainable through the use of traditional OPE [8]. In particular, the mathematical prerequisites for this approach are considerably less stringent compared to fully homomorphic cryptography [11]. To address computational complexity, a lightweight homomorphic encryption tailored for decision tree inference is presented in [12]. While [10, 12] is effective for decision trees, the hardware design for ensembles of trees and their FPGA implementation remains unexplored. High-level synthesis (HLS) tools such as HLS4ML and Conifer are available to generate hardware designs for Xilinx's FPGA [13, 14]. However, these tools are based on Xilinx IP and require Vivado HLS for design generation, making them incompatible with Intel's FPGA. Furthermore, these HLS tools lack APIs for incorporating cryptographic modules. Following [8], significant advances have been made to enhance the security of OPE against chosen-plaintext attacks. Evaluating the adaptation of the latest OPE to decision trees and RFs is crucial. Furthermore, the constraints in selecting between successive and parallel tree evaluations have not been thoroughly investigated.

The above open problems motivate our own exploration of the latest OPE methods, their relevance to privacy preservation within the RF context, the hardware design for trees ensembles, and ultimately the implementation on FPGA. We devise the digital circuit design for the RF model without relying on any IP core, ensuring platform independence, which enables operations on both Xilinx and Intel FPGA. Moreover, it is crucial to present the trade-off between RF inference accuracy and FPGA resource utilization to comprehensively assess the hardware design and its overhead. In this study, we strive to address these aspects. Our contributions are the following: (1) we propose a novel OPE algorithm for the confidential inference on RFs, and (2), we evaluate an FPGA implementation of OPE-encrypted RF on an Intel FPGA and analyze the tradeoffs between inference accuracy, security strength, and FPGA resources.

The remainder of this paper is organized as follows. In Section 2, we provide background on OPE and its application to random forests. In Section 3, the hardware circuit design of encrypted random forest and its synthesis on FPGA is described along with the communication protocols that we have set up to obtain inference on the FPGA. The hardware overhead, numerical results, and discussion of encryption strength are given in Section 4. The paper concludes in Section 5.

2 ENCRYPTION OF RANDOM FOREST

2.1 Order-Preserving Encryption

An order-preserving function f mapping one set $\{0, \ldots, R_p\}$ to another $\{1, \ldots, R_c\}$, where $c > p$, can be uniquely expressed by a selection of p out of c ordered items. R_p and R_c denote the ranges of

plaintexts and ciphertexts, respectively. The encryption function's key consists of two components:

(1) the probability distribution of the ciphertexts $\{0, \ldots, R_c\}$,
(2) the unique mapping $(i, j), i \in \{0, \ldots, R_p\},\ j \in \{1, \ldots, R_c\}$ that generates an exclusive combination of elements from plaintexts and ciphertexts.

For any two combinations of plaintexts and ciphertexts (m_a, c_a) and (m_b, c_b), if $m_b > m_a$, then the condition $c_b > c_a$ must be met for the encryption function f to preserve order.

The OPE scheme of Boldyreva et al. [8] is the one on which most OPE research is based. This symmetric scheme comprises three algorithms (K_{gen}, Enc, Dec). Boldyreva et al. [8] showed that no efficient OPE system can be CPA (Chosen Plaintext Attack) safe, even when the ciphertext space is exponentially larger than the plaintext space. Several OPE techniques are inherently insecure as they leak significant extra information about their plaintexts from a collection of ciphertexts. Order-Revealing Encryption (ORE) [15] addresses this limitation by ensuring that ciphertexts reveal no more information than the ordering of their plaintexts. Specifically, in ORE the ciphertexts are non-numeric, and a comparison function is introduced to compare the ciphers. ORE encompasses three algorithms $(K_{gen}, Enc, Comp)$:

(1) A key generation algorithm K_{gen} that returns a secret key $\mathbb{K}$ based on the security parameter λ.
(2) An encryption algorithm Enc that takes the secret key $\mathbb{K}$ and a plaintext m to return a ciphertext c.
(3) A comparison algorithm $Comp$ that takes two ciphers (c_i, c_j) and returns $\mathbb{b} \in (0, 1)$ to denote whether c_i or c_j is bigger.

The encryption algorithm incorporates a parameter λ that governs the size of the key and ultimately the size of ciphertexts. For a message m of 8 bits, varying the parameter 'λ' results in different sizes of ciphertexts—potentially 16-, 24-, 32-, 64-bits, and so forth. In the context of random tree inference, representing the ciphertexts as non-numeric and implementing the $Comp$ algorithm on an FPGA introduces significant overhead. Consequently, we convert non-numeric ciphers into numeric ones and employ arithmetic comparison operators $(>, \leq)$ to determine the order of the ciphertexts. In Section 3.2 of [15], the authors recommend such conversion, particularly for applications where a numeric ciphertext space is more convenient and order relations can be computed without a "custom" comparison function $Comp$. Furthermore, such a conversion does not degrade the security of the encryption as outlined in [15] (Page 4, Section 1.1, second paragraph). For a full understanding of the $(K_{gen}, Enc, Comp)$ algorithms, readers are referred to [15]. In [10], the OPE of [8] is applied to decision trees.

2.2 Encrypted Decision Nodes

Consider the training dataset $\mathbb{D} = [\mathbb{D}_x, \mathbb{D}_y]$, where $\mathbb{D}_x$ denotes the training samples and $\mathbb{D}_y$ denotes the class labels. In classification, the label of a leaf node corresponds to one of the labels in $\mathbb{D}_y$. Let L be the number of features, and denote the i-th feature as $\mathbb{D}_{x_i}$, $1 \leq i \leq L$. The number of unique values in $\mathbb{D}_y$ represents the number of class labels, denoted as ℓ. Assuming the trained RF model consists of $\mathbb{T}$ decision trees, consider one of the trees with N internal nodes: $I_1, I_2, \ldots, I_N$, and $\mathbf{n}$ terminal or leaf nodes: $T_1, T_2, \ldots, T_n$. The decision rule at I_k is encapsulated in the threshold value Λ_k. The data feature $\mathbb{D}_{x_k}$ at node I_k and the threshold Λ_k define the inequality used in the decision rule of node I_k, expressed as

$$\mathbb{D}_{x_k} < \Lambda_k \text{ or } \mathbb{D}_{x_k} \geq \Lambda_k \tag{1}$$

The architecture of an RF model can be represented as $\mathbb{M}(\mathbb{T})$, where $\mathcal{M}_j(\mathbf{N}_j, \mathbf{n}_j) \in \mathbb{M}$ for $1 \leq j \leq \mathbb{T}$. Here, the index j denotes each decision tree out of $\mathbb{T}$ trees, and the notation $(\mathbf{N}_j, \mathbf{n}_j)$ represents the number of internal nodes and leaves of the j^{th} decision tree. RF inference involves traversing each decision tree from the root node to one of the leaf nodes. The internal nodes visited during this traversal are those defined by a chain of inequalities similar to (1). The inferred label is that of the leaf node at the end of the tree traversal, representing one of the class labels $b_j \in [0, \ell - 1]$ of the dataset for the j^{th} decision tree. The final outcome is determined by majority voting on the predictions of each decision tree:[1] $b =$ majority voting$(b_1, b_2, \ldots, b_j, b_{j+1}, \ldots, b_{\mathbb{T}})$

During encryption, the feature and threshold associated with a specific decision rule at a given tree node are used. Encrypting the RF involves encrypting the computations at the internal nodes of each decision tree. Let $\mathbf{E}_{\mathbb{K}}$ represent the OPE operator as described in [9]. The encrypted equivalents of the RF inequalities are then expressed as:

$$\mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_k}) < \mathbf{E}_{\mathbb{K}}(\Lambda_k) \textit{ or } \mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_k}) \geq \mathbf{E}_{\mathbb{K}}(\Lambda_k) \tag{2}$$

Note that in the above inequalities, the order $(<, \geq)$ of $\mathbb{D}_{x_k}$ with respect to Λ_k is preserved as in Eq. (1), i.e., $\mathbb{D}_{x_k} - \Lambda_k = \mathbf{D}_{\mathbb{K}}(\mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_k}) - \mathbf{E}_{\mathbb{K}}(\Lambda_k))$, where $\mathbf{D}_{\mathbb{K}}$ is the decryption operator using the $\mathbb{K}$ secret key. The secure representation of the RF must produce the correct inference result in encrypted form. Consequently, the labels of the leaf nodes T_k are also encrypted as $\mathbf{E}_{\mathbb{K}}(T_k)$. The inferred class or label is subsequently determined using the private key $\mathbb{K}$ as follows:

$$T_k = \mathbf{D}_{\mathbb{K}}(\mathbf{E}_{\mathbb{K}}(T_k)) \tag{3}$$

The majority voting rule is given by

$$b = \mathbf{D}_{\mathbb{K}}\left[\text{majority voting}\{\mathbf{E}_{\mathbb{K}}(b_1), \mathbf{E}_{\mathbb{K}}(b_2), \ldots, \mathbf{E}_{\mathbb{K}}(b_{\mathbb{T}})\}\right] \tag{4}$$

Training RF is carried out with plaintexts $\mathbb{D} = [\mathbb{D}_x, \mathbb{D}_y]$. After the RF is trained, OPE is applied as given in Eq. (2). The encryption of one of the trees $\mathbf{E}_{\mathbb{K}}(\mathcal{M}_1(N_1, n_1))$ in RF is shown in Figure 1. Similar encryption is applied for all other trees $\mathbf{E}_{\mathbb{K}}\left(\mathcal{M}_j(N_j, n_j)\right)$, $2 \leq j \leq \mathbb{T}$.

3 FPGA DESIGN OF RANDOM FOREST

In this investigation, Verilog is used as the hardware description language. Our FPGA implementation is categorized into two types of digital logic circuits: combinational and sequential. The combinational design of [12] and the sequential design of [10] are restricted to decision trees. In this paper, they are extended to RFs.

3.1 Edge Cloud Simulation

In the experimental testbed shown in Figure 2, an open source Python library *sklearn* [16] is used to train the RF model. Training and validation have been carried out on a compute intensive Amazon *m5d.2xlarge* EC2 virtual machine (VM). For OPE (modified ORE

[1]The notation λ refers to the security parameter of the ORE cryptography while Λ refers to the threshold values of the RF model. Also, the notation $\mathbb{b}$ refers to outcome of *Comp* operator of ORE while b refers to the inference outcome of the RF model.

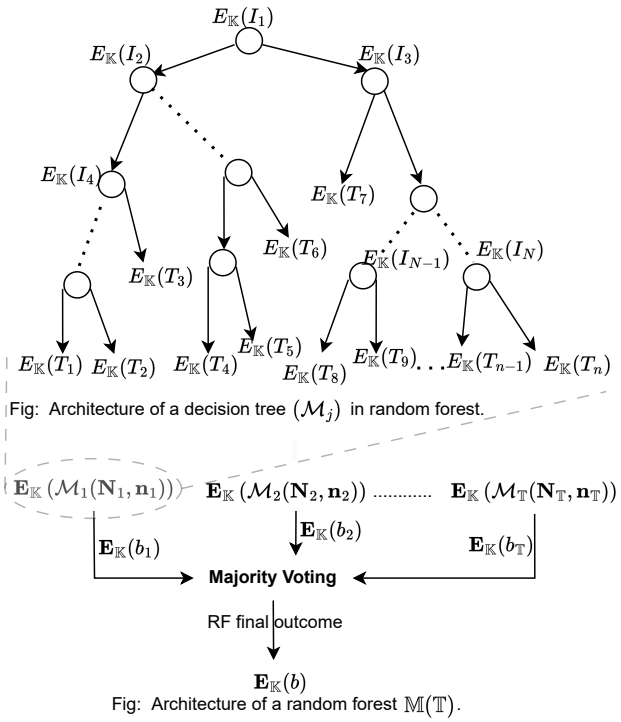

Figure 1: Encrypted decision tree in random forest: At each node, a comparison inequality is made between a dataset feature $\mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_k})$ and a threshold $\mathbf{E}_{\mathbb{K}}(\Lambda_k)$. If the outcome of the inequality is true, the evaluation proceeds on the left side of the node; otherwise, it proceeds on the right side. The lower part illustrates the outcome of each decision tree during encrypted Random Forest inference.

as given in Section 2.1), we have used the open source code available at [17]. Once the training is completed, the decision rules from the root node to each leaf node are retrieved. These rules are then transformed into Verilog source code. For such a transformation, we use the tool described in [12]. The Quartus Prime Design Suite is used for the RTL synthesis of the RF's Verilog source code and the generation of the SRAM-object file (sof). For the demonstration, we have used the well-known MNIST dataset [18], a collection of grayscale images of handwritten digits from 0 to 9, ($\ell = 10$). The features of the dataset are the pixels of the grayscale image with size $L = 28 \times 28 = 784$. For the FPGA implementation, we have selected the DE-10 Standard *SCSXFC6D6F31CN* FPGA board [19]. The FPGA acts as an edge device connected to the host machine through the UART port. After Quartus logic synthesis, the SRAM object file (sof) is sent over the USB port and loaded into the FPGA. Two inference experiments, *Expt1* and *Expt2* are carried out. In *Expt1*, the model is implemented on FPGA without any encryption, while in *Expt2*, the nodes and leaves of the RF model are encrypted as described in Section 2.2.

We model a two-party transaction where *Alice* possesses an RF model, and *Bob* holds the inferential data, as shown in Figure 2. Both parties have mutually agreed upon a specific OPE key $\mathbb{K}$. Upon completion of the inference run, *Bob* receives the encrypted class label from the RF, and $\mathbb{K}$ is applied to recover the true label.

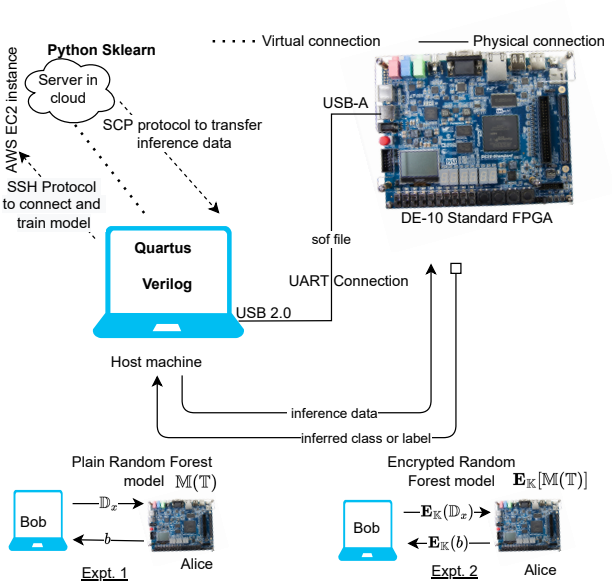

Figure 2: Experimental testbed of the edge-cloud environment.

3.2 Communication Protocol

We carry out all communication using the serial UART protocol. The connection diagram is shown in Figure 2. The MNIST pixels $\mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_i}), 1 \leq i \leq L$ are transmitted from host machine to the FPGA while the RF's inference result $\mathbf{E}_{\mathbb{K}}(b)$ is sent by the FPGA to the host machine through *FT232RL*. The MNIST data samples have pixel values ≤ 255. This causes the Λ thresholds of the decision node to be ≤ 255 too. With that specification, an 8-bit $(255 = 2^8 - 1)$ representation of the decision tree model and inference data is adequate. However, the ciphers of nodes, leaves, and inference data may be more than 255. The cipher size $(|c|)$ is determined by the OPE parameters λ given in [17].

For serial communication, we employ an 8−bit word representation. As a result, a ciphertext c of more than 8−bit width is subdivided into packets of a 8bits each. The number of 8− bit words N_w for a ciphertext is then calculated as:

$$N_w = \left\lceil \lfloor \frac{\lfloor log_2(c) \rfloor}{8} \rfloor + 1 \right\rceil \tag{5}$$

The width of RAM that carries the ciphertext c with respect to $\mathbf{E}_{\mathbb{K}}(\mathbb{D}_x)$ has a greater width than the RAM that holds the plaintext messages m corresponding to $\mathbb{D}_x$. The same holds for the Λ thresholds and their ciphertexts $\mathbf{E}_{\mathbb{K}}(\Lambda)$'s. We have chosen three values of the ORE security parameter λ so that the ciphertext size $(|c|)$ is $\{24, 32, 64\}$ bits.

3.3 Circuit Design of Random Forest

In [12], the combinational logic design of an encrypted decision tree is explored. The control flow structure of a tree in any programming language typically involves a sequence of nested "if-then-else" statements concluding with a clause that represents the class label outcome of the leaf node. Such conditional statements are equivalent to the inequality relationships presented in (1) and (2). These inequalities, corresponding to the nodes between the root and any

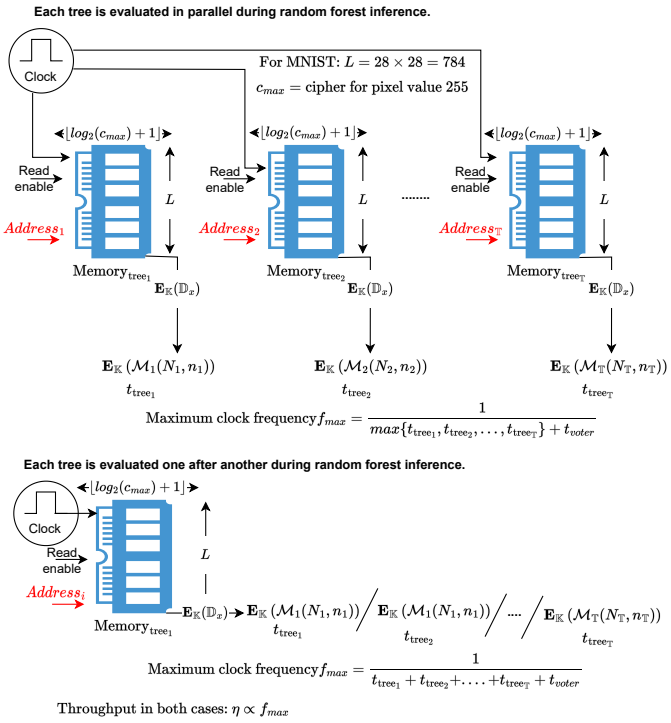

Figure 3: FPGA memory map showing parallel vs successive processing of trees during the random forest inferencing.

given leaf, result in a Boolean function that mirrors the structure of the tree as shown in Figure 1. For example,

$$\mathbf{E}_{\mathbb{K}}(T_1) = \mathbf{E}_{\mathbb{K}}(I_1)\ \&\ \mathbf{E}_{\mathbb{K}}(I_2)\ \&\ \mathbf{E}_{\mathbb{K}}(I_4)\ \&\ \ldots$$
$$\mathbf{E}_{\mathbb{K}}(T_3) = \mathbf{E}_{\mathbb{K}}(I_1)\ \&\ \mathbf{E}_{\mathbb{K}}(I_2)\ \&\ \overline{\mathbf{E}_{\mathbb{K}}(I_4)}$$

where "&" represents the logical AND operation.

In [10], an alternative sequential logic design is suggested where the transition graph of its finite-state machine (FSM) mirrors that of the decision tree (DT), with each tree node representing one FSM state. During each clock cycle, one of the FSM states is assessed by examining a single inequality relation, as described in (2). In the hardware, this comparison is realized through a comparator and a multiplexer. The comparison attributes, namely the dataset feature $\mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_i})$ for each decision tree node, are stored in a RAM. Inference data samples are retrieved from the RAM and input to the root node I_1 of each decision tree. Subsequent states are determined in the following clock cycle after computing the inequality operation of I_1. This iterative process continues until the evaluation reaches a leaf state, i.e., an encrypted terminal node $\mathbf{E}_{\mathbb{K}}(T_i)$. Upon reaching a leaf state, the class/label of $\mathbf{E}_{\mathbb{K}}(T_i)$ is extracted and returned as the inference result of that particular DT. The same processing applies to other DT's of the RF model.

This paper employs similar combinational [12] and sequential [10] logic circuits to implement each tree. The width (or number of bits) of the RAM is set to match the size of the maximum value of the ciphertext $c_{\max}$. In OPE, this ciphertext corresponds to the maximum value of the MNIST pixel, which is 255.

3.4 Parallel vs. Sequential Tree Evaluation

The length (or the number of addresses) of the RAM equals the number of features in the dataset, namely, $L = 28 \times 28 = 784$ for the MNIST dataset. The visual representation is shown in Figure 3. Two designs of the memory map are presented. In the first architecture, $\mathbb{T}$ copies of RAM are utilized, with one RAM allocated to inferencing one tree in the RF. All the copies of RAM are triggered by the same clock frequency. But the memory address of each RAM is different. For example, assume $\mathbb{T} = 3$, and consider the following computation at the root node:

$$[\mathbf{E}_{\mathbb{K}}(I_1)]_{tree_1} : \mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_i}) < \mathbf{E}_{\mathbb{K}}(\Lambda_i)$$
$$[\mathbf{E}_{\mathbb{K}}(I_1)]_{tree_2} : \mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_j}) < \mathbf{E}_{\mathbb{K}}(\Lambda_j)$$
$$[\mathbf{E}_{\mathbb{K}}(I_1)]_{tree_3} : \mathbf{E}_{\mathbb{K}}(\mathbb{D}_{x_k}) < \mathbf{E}_{\mathbb{K}}(\Lambda_k)$$

In this context, the indices i, j, k denote the i^{th}, j^{th}, and k^{th} pixel values within the range of 0 to 255. These indices correspond to RAM addresses $\{Address_1, Address_2, Address_3\}$ highlighted in red in Figure 3. Similar mapping extends to other nodes of each tree. Given the constraint of only one memory read per clock cycle in the specified FPGA device, it becomes imperative to allocate a dedicated memory for each tree to enable concurrent processing of multiple trees. Consequently, the RAM has been scaled for each tree to accommodate such requirement. In this scenario, the FPGA resources in terms of flip-flops, logic blocks, and DSP slices increase with the number of trees. As each tree is computed in parallel, the utilization of *AND* gates in the combinational design and *Comparators/Multiplexers* in the sequential design scales up linearly with respect to $\mathbb{T}$.[2]

The second architecture employs only one RAM, and the trees are processed sequentially. After one tree produces the inference b_j, it is stored in a register, and the processing of the next tree commences, as illustrated in Figure 3. Once all the trees have completed their inferences, the final outcome is determined using the majority voting mechanism. Clearly, the second architecture would require much more time to complete the inference. The maximum clock frequency at which the overall design can operate is significantly lower in the second architecture than in the first. Additionally, this clock frequency degrades when scaling up the number of trees. Ultimately, the clock frequency determines the inference throughput.

Clearly, there is a trade-off between the throughput and FPGA resource utilization. *In this work, an edge-cloud environment is considered, in which acceleration is prioritized. Consequently, we choose the first design, which involves concurrently computing trees in a RF.* In either design, inference data samples are read from RAM and fed into the tree root node $\mathbf{E}_{\mathbb{K}}(I_1)$. The cipher thresholds $\mathbf{E}_{\mathbb{K}}(\Lambda)$'s are hardcoded in the Verilog script for both the combinational and sequential design. The subsequent nodes, $\mathbf{E}_{\mathbb{K}}(I_2)$ or $\mathbf{E}_{\mathbb{K}}(I_3)$, are evaluated based on the query outcome of the previous node. Once the leaf is reached, the class/label of $\mathbf{E}_{\mathbb{K}}(T_i)$ is returned as the inference result of that particular tree. The majority voting is applied over all the tree's outcomes through the combinational circuit; an example for $\mathbb{T} = 3$ is shown in Fig 4. The circuit diagram is constructed

[2]A pipelined design may be employed instead of scaling the RAM. Indeed, memory access could be pipelined with the comparison operations ($\leq$, $>$). The High-Level Synthesis (HLS) tool, integrated with OPE, can be employed to generate the optimized design. We expect to report on the pipelined design in a forthcoming publication.

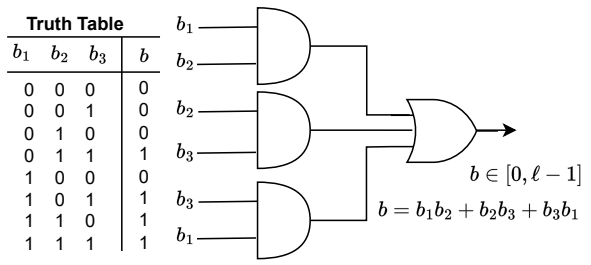

Figure 4: Majority voting over each $\mathcal{M}_j(N_j, n_j)$'s outcome b_j's to produce b, the final inference outcome of random forest model $\mathbb{M}(\mathbb{T})$.

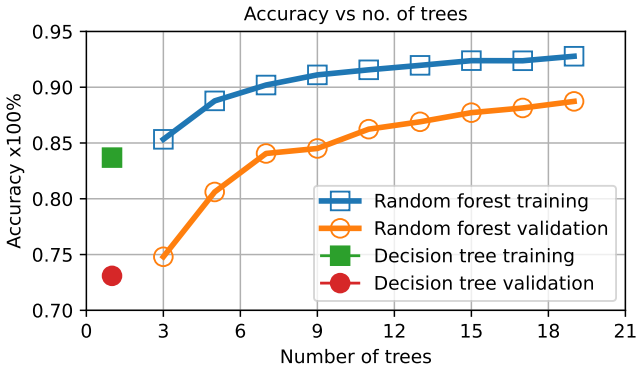

Figure 5: Accuracy of a decision tree (DT) and random forest (RF) with various values of $\mathbb{T}$. For each tree, the limit is set such that $N_j \leq 600, n_j \leq 300, 1 \leq j \leq \mathbb{T}$, and maximum 20 nodes between root and a leaf across all trees. *The accuracies of plain and encrypted RF/DT are the same.*

based on the truth table, representing all possible combinations for various outputs of each tree during RF inference. Consider the case where $b_1 = b_2 = \ell$, then the majority voting $b = \ell(1 + b_3 + b_3) = \ell$, thus the Boolean expression produces the correct inferential outcome. Similarly, one could test other scenarios of the truth table. For larger number of trees, one may consider adopting the hardware design using multiplexers.

4 EXPERIMENTAL EVALUATION

4.1 FPGA Resource Analysis

We evaluated the inference accuracy across various decision tree architectures, as depicted in Figure 5. The chosen architecture features $N_j = 600$ nodes and $n_j = 300$ leaves, with a total of $\mathbb{T} = 3$ trees, where $1 \leq j \leq \mathbb{T}$.[3] This parameter selection resulted in an accuracy of 86.12% on the validation dataset. Two inference experiments, denoted as *Expt1* and *Expt2*, were conducted as explained in Section 3.1. A comparison of FPGA resources is presented in Tables 1 and 2. The percentage utilization in Tables 1 and 2 represents the fraction of total ALMs (adaptive logic modules), LABs (logic array blocks), and FFs (dedicated primary registers) of the DE-10 standard FPGA used in the RF implementation. The FPGA clock frequency is set to 50 MHz. As anticipated, the resources consumed by the encrypted RF model exceed those of the plain one. Of particular interest is the use of LABs and FFs in the encrypted case due to additional storage requirements. Such use may contribute to increased power consumption during additional READ operations corresponding to large ciphertexts. The functionality and performance of the design are determined by the slack setup/hold time, both of which are positive in our case. Three tests were conducted for the encrypted scenario, employing different ciphertext sizes by varying the OPE security parameter λ. This resulted in ciphertext lengths of {18, 32, and 64}-bits, corresponding to $N_w = \{3, 4, \text{ and } 8\}$, respectively. During Quartus synthesis of the behavioral Verilog code corresponding to the RF model, the resource consumption of ALMs and LABs exceeded 100%, indicating that the available FPGA resources for the specified FPGA were insufficient to store and operate the RF design. Consequently, the synthesis failed to complete the run, and timing analysis results were not returned by Quartus. For such designs, the outcomes are marked as '×' in Tables 1 and 2. An alternative approach to resolve the '×' cases is to sacrifice throughput and evaluate the trees serially (Figure 3).

Surprisingly, the sequential design outperformed the combinational design as it successfully completed the synthesis of the RF design with 32-bit ciphertexts. Because the sequential design is

Table 1: FPGA resource comparison for random forest inference with $\mathbb{T} = 3$ in combinational design (design circuit available in [12]). ALMs: Adaptive logic modules, LABs: Logic array blocks, ALUTs: Combinational adaptive look-up tables, FFs: Dedicated primary registers, F_{max}: Maximum attainable clock frequency, Setup: Slack time for clock setup, Hold: Slack time for clock hold.

Attributes	Unencrypted (Fig. 2 *Expt1*)	OPE as per Section 2.1 (Fig. 2 *Expt2*)		
		$N_w = 8$ ($\|c\| = 64$)	$N_w = 4$ ($\|c\| = 32$)	$N_w = 3$ ($\|c\| = 24$)
ALMs (%)	9	136	122	75
LABs (%)	13	136	123	97
ALUTs	5795	63509	54033	28670
FFs (%)	4	90	87	66
$F_{max}(MHz)$	68.92	×	×	62.11
Setup (ns)	5.491	×	×	3.899
Hold (ns)	0.372	×	×	0.344

Table 2: FPGA resource comparison for random forest inference with $\mathbb{T} = 3$ in sequential design (design circuit available in [10]).

Attributes	Unencrypted (Fig. 2 *Expt1*)	OPE as per Section 2.1 (Fig. 2 *Expt2*)		
		$N_w = 8$ ($\|c\| = 64$)	$N_w = 4$ ($\|c\| = 32$)	$N_w = 3$ ($\|c\| = 24$)
ALMs (%)	15	140	49	41
LABs (%)	23	142	68	55
ALUTs	9960	69129	31221	27547
FFs (%)	4	89	30	25
$F_{max}(MHz)$	71.73	×	61.7	71.2
Setup (ns)	5.578	×	3.793	4.987
Hold (ns)	0.371	×	0.365	0.368

[3]For illustrative purposes, we have used the MNIST dataset and implemented an RF with three DTs. Our methodology is adaptable to any dataset, with the number of DTs determined through hyperparameter tuning, as shown in Fig. 5, provided that the FPGA resources can accommodate the specified number of trees.

triggered at the clock edge, the computation in just one state of the FSM is completed in one clock cycle. The computation performed in each clock cycle can be precisely regulated using the clock. Such clock control is not accessible in the combinational design, and the entire tree computation is triggered by any change in the module inputs. Consequently, significant FPGA resource usage occurs in the combinational design. Additionally, the value of F_{max} indicates that the sequential design has the ability to operate at a higher clock frequency. Therefore, the sequential design is favored for confidential inference.

4.2 Discussion

In OPE, the size of ciphertexts plays a pivotal role in fortifying cryptographic strength. Larger ciphertexts size $|c|$ adds complexity, rendering traditional cryptanalysis techniques more challenging for potential adversaries. This increased complexity offers protection against frequency analysis, where the distribution of plaintext values could be exploited. Moreover, longer ciphertexts bolster resistance against dictionary attacks, as the computational burden of precomputing exhaustive tables becomes impractical. The expanded search space provided by larger ciphertexts enhances security by making it more harder for attackers to guess or infer plaintext values based on partial information. It is therefore advisable to utilize a considerably larger value of $|c|$ that can fit within the FPGA.

The trend illustrated in Fig. 5 reveals that augmenting the number of trees, denoted as $\mathbb{T}$, leads to increased accuracy. However, incorporating numerous trees may prove inadequate for FPGA resources. According to Table 2, the maximum 32-bit ciphertext size can be accommodated, but this comes at the cost of utilizing approximately half of the ALMs and LABs. Introducing additional trees, such as $\mathbb{T} = 5, 7, 9, 11, ..$, becomes impractical. The demonstration illustrating this is presented in Table 3. Despite compromising the security strength, i.e., setting the cipher size to $|c| = 16$ bits, and using only 9 trees, more than half of the resources are already consumed. The model developer is expected to take into account the desired RF's inference accuracy while adhering to FPGA resource constraints along with security strength.

5 CONCLUSIONS

This paper presented the first use of OPE for the confidential inference in an RF model. Parallel and sequential tree evaluations are used to achieve a resource-efficient FPGA implementation of the encrypted RF. Additionally, a serial communication protocol is specifically designed to streamline hardware-accelerated inference in edge-cloud environments. In an upcoming publication, we will develop various attacks on the OPE of RFs and assess its resilience using standard cybersecurity benchmarks.

Table 3: FPGA resource comparison for inferencing over different random forest architecture with $|c| = 16$-bits (or $N_w = 2$) in sequential design (design circuit available in [10]).

Attributes	OPE as per Section 2.1 (Fig. 2 *Expt2*)			
	$\mathbb{T} = 3$	$\mathbb{T} = 5$	$\mathbb{T} = 7$	$\mathbb{T} = 9$
	Acc ≈ 75%	Acc ≈ 80%	Acc ≈ 84%	Acc ≈ 85%
ALMs (%)	25	47	58	67
LABs (%)	33	52	72	79
ALUTs	14327	23125	29134	35123
FFs (%)	17	29	42	53
$F_{max}(MHz)$	71.4	70.6	67.7	66.3
Setup (ns)	5.182	5.053	4.019	4.002
Hold (ns)	0.37	0.362	0.358	0.342

ACKNOWLEDGMENT

This research has been conducted at Khalifa University with financial support provided by the Technology Innovation Institute (TII), Abu Dhabi, UAE, under contract TII/CRP/2036/2020. The authors are grateful to Elena Kirshanova, Nitin Satpute, and Jinson Thomas, all from TII, for enlightening discussions and helpful comments. Rupesh Raj Karn is now with New York University, Abu Dhabi, UAE.

REFERENCES

[1] Qianying Liao, Bruno Cabral, João Paulo Fernandes, and Nuno Lourenço. Herb: Privacy-preserving random forest with partially homomorphic encryption. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2022.
[2] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. Privacy preserving vertical federated learning for tree-based models. *arXiv preprint arXiv:2008.06170*, 2020.
[3] Zhuoran Ma, Jianfeng Ma, Yinbin Miao, and Ximeng Liu. Privacy-preserving and high-accurate outsourced disease predictor on random forest. *Information Sciences*, 496:225–241, 2019.
[4] Nicholas Carlini et. al. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019.
[5] Zhanglong Ji, Zachary C Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.
[6] Asma Aloufi et. al. Blindfolded evaluation of random forests with multi-key homomorphic encryption. *IEEE Transactions on Dependable and Secure Computing*, 18(4):1821–1835, 2019.
[7] David J Wu, Tony Feng, Michael Naehrig, and Kristin Lauter. Privately evaluating decision trees and random forests. *Cryptology ePrint Archive*, 2015.
[8] Alexandra Boldyreva et. al. Order-preserving symmetric encryption. In *Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28*, pages 224–241. Springer, 2009.
[9] Kevin Lewi and David J Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1167–1178, 2016.
[10] Rupesh Raj Karn, Kashif Nawaz, and Ibrahim M Elfadel. Securing decision tree inference using order-preserving cryptography. In *5th IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS 2023)*, pages 1–5. IEEE, 2023.
[11] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.
[12] Rupesh Raj Karn and Ibrahim M Elfadel. Confidential inference in decision trees: FPGA design and implementation. In *30th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC 2022)*, pages 1–6. IEEE, 2022.
[13] Sioni Summers, Giuseppe Di Guglielmo, Javier Duarte, Philip Harris, Duc Hoang, Sergo Jindariani, Edward Kreinar, Vladimir Loncar, Jennifer Ngadiuba, Maurizio Pierini, et al. Fast inference of boosted decision trees in fpgas for particle physics. *Journal of Instrumentation*, 15(05):P05026, 2020.
[14] Rafal Kulaga and Marek Gorgon. FPGA implementation of decision trees and tree ensembles for character recognition in vivado hls. *Image Processing & Communications*, 19(2-3):71, 2014.
[15] Nathan Chenette et. al. Practical order-revealing encryption with limited leakage. In *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016*, pages 474–493. Springer, 2016.
[16] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
[17] Kevin Lewi David Wu. Fastore.
[18] Yann LeCun et. al. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[19] DE-10 standard FPGA. http://de10-standard.terasic.com/.