UC Secure Private Branching Program and Decision Tree Evaluation

Keyu Ji
jikeyu@zju.edu.cn
Zhejiang University

Bingsheng Zhang
bingsheng@zju.edu.cn
Zhejiang University

Tianpei Lu
21921147@zju.edu.cn
Zhejiang University

Lichun Li
lichun.llc@antgroup.com
Ant Group

Kui Ren
kuiren@zju.edu.cn
Zhejiang University

Abstract
Branching program (BP) is a DAG-based non-uniform computational model for L/poly class. It has been widely used in formal verification, logic synthesis, and data analysis. As a special BP, a decision tree is a popular machine learning classifier for its effectiveness and simplicity. In this work, we propose a UC-secure efficient multi-party computation platform for outsourced branching program and/or decision tree evaluation. We construct a constant-round protocol and a poly-round protocol. In particular, the overall (online + offline) communication cost of our poly-round protocol is $O(d(\ell + \log m + \log n))$ and its round complexity is $2d - 1$, where $m$ is the DAG size, $n$ is the number of features, $\ell$ is the feature length, and $d$ is the longest path length. To enable efficient oblivious hopping among the DAG nodes, we propose a lightweight 1-out-of-N shared OT protocol with logarithmic communication in both online and offline phase. This partial result may be of independent interest to some other cryptographic protocols. Our benchmark shows, compared with the state-of-the-arts, the proposed constant-round protocol is up to $10X$ faster in the WAN setting, while the proposed poly-round protocol is up to $15X$ faster in the LAN setting.

1 Introduction

Branching program (BP) or binary decision diagram is a nonuniform computational model for L/poly class. The computation is specified by a directed acyclic graph (DAG) with a unique source node and several sink nodes; an evaluation is usually performed by traverse from the source node to a sink node. BP has been widely used in formal verification, logic synthesis and data analysis, etc. In particular, decision tree is a special case of BP, known for its effectiveness and simplicity as a machine learning classifier with a number of useful applications, including credit-risk assessment, spam classification, medical diagnosis.

Privacy concerns often raise, when sensitive information are involved. In the past decades, the privacy-preserving BP and decision tree evaluation problem has been extensive studied in the literature [2, 3, 7, 15, 16, 18, 21, 22, 26]. These works can be divided into two main categories based on protocol round complexity: (i) constant-round solutions [3, 7, 15, 21, 26], and (ii) polynomial-round solutions [11, 16, 20, 22]. As summaries in [18], a typical constant-round solution consists of three functional modules: (a) private feature selection, (b) secure comparison, and (c) oblivious path evaluation. Each step can be realized by either garbled circuit or homomorphic encryption based protocols. The overall protocol usually needs to obliviously evaluate each decision node of the DAG for privacy preservation; therefore, they are suitable for BPs and decision trees with small DAG size, say less than $2^{20}$. On the other hand, polynomial-round solutions can bypass this limitation by obliviously hopping along a DAG path according to the outcome of previous decision nodes. This is known as oblivious access index (OAI) [22], which can be realized by either OT or ORAM. The OT-based OAI private decision tree evaluation protocol proposed in [22] takes linear communication (in tree size, $m$) and $4d$ rounds. When OAI is realized by Circuit ORAM [24], the online communication complexity can be reduced to $O(d^3)$, but it takes up to $O(d^2)$ rounds. Moreover, the ORAM initialization phase is very slow for large tree size and/or feature numbers. For instance, it could take 20 days to insert $2^{16}$ elements of 512 bits each [14].

The best polynomial-round solution is recently proposed by Ma et al. [20]. It reduces the online communication cost to $O(d)$ using key management and conditional OT. However, prior to each evaluation, the model owner has to prepare and share a one-time encoding of the tree to the client, which leads to linear communication in the offline phase. Meanwhile, the protocol proposed by [20] can be modified to fit the outsourcing setting, where the model owner and the data owner just need to share their private input to the computing servers without heavily involved in the evaluation process. This setting enables the usage scenarios when the features are spited among multiple clients, and it is friendly to mobile devices with low-computation resources, such as IoT sensors. However, their outsourcing solution [20] needs to pad the
decision tree to a complete tree for privacy preservation, and it costs $O(2^d)$ communication to refresh the shared decision tree in the offline phase of each evaluation. In addition, their solution does not naturally support BP evaluation.

1.1 Our approach

In this work, we investigate the outsourced private branch-
ing program and decision tree evaluation problem. We first propose a new 1-out-of-$N$ shared OT protocol with logarithmic communication. In a shared OT, given a vector of shared messages $\mathbf{x} := (x_0, \ldots, x_{N-1})$ and an shared index $i \in \mathbb{Z}_N$, the MPC parties can jointly obtain $x_i$ in the shared form without revealing $i$. Our approach follows the line of research initiated by Boyle et al. [5], which introduces the distributed point function (DPF). DPF enables an efficient two-server PI protocol, where two servers hold the same set of messages $x$ and the client wants to obliviously fetch $x_i$. Namely, the client first generates a pair of DPF keys encoding a point function $f_i(x)$, which has only one non-zero output, 1, when the input is $i$. The client then distributes the DPF keys to the two servers, and the servers jointly evaluate and return $x_i := \sum_{j=0}^{N-1} f_i(j) \cdot x_j$. Later, Doerner et al. [12] adopt DPF in the MPC setting to achieve ORAM. In [12], both servers $S_1$ and $S_2$ hold encrypted messages $\bar{x}_j := x_j \oplus \text{PRF}_k(j)$, $j \in \mathbb{Z}_N$, where $k$ is shared between them. For a given shared index $i \in \mathbb{Z}_N$, $S_1$ and $S_2$ first generates the DPF keys for $f_i(x)$ via MPC. After obtaining the shared $\bar{x}_i$, $S_1$ and $S_2$ then needs to obliviously evaluate $\text{PRF}_k(i)$ via MPC to decrypt $x_i$. Therefore, the entire process is time-consuming.

In this work, we eliminate the needs of aforementioned two costly MPC operations by introducing more servers. Our platform utilizes four servers $S_1, \ldots, S_4$. Suppose $S_1$ and $S_2$ needs to evaluate the DPF on $i$, where $i$ is additively shared among four servers. To avoid MPC generation of DPF keys, we let a third server (an non-evaluator of this DPF), say $S_3$, to obliviously evaluate $\text{PRF}_k(i)$ via secure comparison based on a secure comparison result. It can be realized by a DCF evaluation and then a shared multiplication, but it would take 2 rounds. To reduce round complexity, we divide the four servers into two groups. Each group independently evaluates a DCF to perform secure comparison between the corresponding threshold and feature in a parallel. Subsequently, the shared multiplication can be reduced to a scalar product which can be evaluated locally without further communication. Once a sink node is reached, the servers would obliviously evaluate the dummy node (repeatedly) until the protocol reaches $d$ total steps. The classification values of all nodes in the evaluation path are summed to the final result.

Our constant-round solution. We construct a 3-round private decision tree evaluation protocol, using the proposed 1-out-of-$N$ shared OT protocol as a building block. We assume the model and features are already shared among the four servers. Note that the model needs to be padded to a complete tree to avoid privacy leakage. In the first round, the servers obliviously select corresponding features for all decision nodes. In the second round, for each decision node, a secure comparison is performed using distributed comparison function (DCF) [4]. More specifically, $S_1$ plays the role of DCF key generator while $S_1$ and $S_2$ play the role of DCF evaluators. In the offline phase, $S_4$ precomputes the DCF keys and distribute them to $S_1$ and $S_2$. In the online phase, the servers mask the difference of its threshold and feature, and open it to $S_1$ and $S_2$. They then jointly evaluate DCF to securely compare the corresponding feature with the threshold. When the feature is less than the threshold, $S_1$ and $S_2$ obliviously set the left out-going edge cost of the decision node to 0 and the right out-going edge cost to a random value; vice versa. In the third round, for each leaf node of the decision tree, $S_1$ and $S_2$ sum up the edge costs along the path to get its path cost. They then cyclic shift the vector of path costs of all the leaf nodes together with the corresponding classification values. After that, $S_1$ and $S_2$ open the shifted path costs and re-share the shifted classification values to $S_3$ and $S_4$. They output the classification value of the leaf node whose path cost is 0 as the evaluation result to the receiver.
Table 1: Performance comparison: \( m \) is the DAG(or decision tree) size, \( m_1 \) is the number of decision nodes, \( m_2 \) is the DAG size after depth-padding, \( m_3 \) is the number of decision nodes in padded tree, \( n \) is the number of features. \( N \) is the number of model owners, \( t \) is the bit-length of feature and classification value, \( \lambda_3 \) is the size of symmetric ciphertext \((= 128)\), \( \lambda_2 \) is the size of ElGamal ciphertext \((= 514 \text{ for 40-bit security})\), \( \lambda_4 \) is the size of Paillier ciphertext \((= 4096 \text{ for 40-bit security})\), \( \lambda_5 \) is the size of AES key \((= 128)\).

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Communication</th>
<th>Outsourcing</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2] (GGG)</td>
<td>( (n + m_1) \log n + 2m_1 \log m_1 - n + 2 + 2m_1 {(\lambda_1 + \log(\bar{m}_1 + 1)) } )</td>
<td>( n(2\lambda_1 + 1) )</td>
</tr>
<tr>
<td>[21] (HHH)</td>
<td>-</td>
<td>( (m_1 + n)(\ell + 2(\log \ell + 1) + n)\lambda_2 )</td>
</tr>
<tr>
<td>[18] (GGH)</td>
<td>( (n + m_1) \log n + 2m_1 \log m_1 - n + 2 + 2m_1 {\lambda_1 } )</td>
<td>( n(2\lambda_1 + 1) + (3m_1 + 2)\lambda_2 )</td>
</tr>
<tr>
<td>[18] (HGH)</td>
<td>( 5m_1\lambda_1 )</td>
<td>( (m_1 + n)\lambda_3 + m_1(2\lambda_1 + 1) + (3m_1 + 2)\lambda_2 )</td>
</tr>
<tr>
<td>[1]</td>
<td>-</td>
<td>( n\lambda_1 + \lambda_5 + \lambda_6 + (2N + 5)\lambda_7 )</td>
</tr>
<tr>
<td>Ours (constant)</td>
<td>( 4 \cdot 2^{\ell}(\log n + \log \ell + d)\lambda_4 )</td>
<td>( 2^{\ell}(3\log n + 3d + 7\ell) )</td>
</tr>
<tr>
<td>[11]</td>
<td>( 6(2^{\ell} n + d(3\ell - \log \ell - 2) + 2^d - 1) )</td>
<td>( 4(2^{\ell} n + d(3\ell - \log \ell - 2) + 2^d - 1) )</td>
</tr>
<tr>
<td>[27]</td>
<td>( 6((2^{\ell} n + 4)d - 5) )</td>
<td>( 4((2^{\ell} n + 4)d - 5) )</td>
</tr>
<tr>
<td>[19]</td>
<td>( 6(2^{\ell} - 1)d )</td>
<td>( 3 \cdot 2^{\ell - 1}\lambda_4 + 4(2^{\ell - 1}) \ell )</td>
</tr>
<tr>
<td>[22] (OT)</td>
<td>( 6d\lambda_1 )</td>
<td>( d(m_1 + n)\ell + 2(\log m + \log n)\lambda_1 )</td>
</tr>
<tr>
<td>[20] (complete)</td>
<td>( 2^{\ell}(\log n + \log \ell + d) )</td>
<td>( d(4\lambda_4 + n\ell + (7\ell + 8)\lambda_1) )</td>
</tr>
<tr>
<td>[20] (sparse)</td>
<td>( m(\ell + \log n + \lambda_5 + 3d) )</td>
<td>( d(4\lambda_4 + n\ell + (7\ell + 8)\lambda_1 + 8) )</td>
</tr>
<tr>
<td>Ours (poly-round)</td>
<td>( 4d(\log n + m + \log \ell)\lambda_4 )</td>
<td>( 12d(2\log m + \log n + \ell) )</td>
</tr>
</tbody>
</table>

**Performance.** Table 1 shows the communication and round complexity comparison between our scheme and the related works. The schemes that supports outsourcing are marked with \( \bullet \). \( m \) is the DAG size, \( m_1 \) is the number of decision nodes, \( m_2 \) is the DAG size after depth-padding, \( m_3 \) is the number of decision nodes in padded tree, \( n \) is the number of features, \( t \) is the bit-length of feature and classification value. We emphasize that the concrete security parameters vary a lot among different schemes, and we use \( \lambda_1, \ldots, \lambda_8 \) to differentiate them. For instance, \( \lambda_4 \) refers to the ciphertext size of Paillier encryption, which is 4096 bits for 40-bit security; whereas, the security parameter \( \lambda_8 \) is the 128-bit AES key size in our schemes. Note that some works (marked with **), e.g., [1,19] do not protect the feature indices from the servers.

Our constant-round protocol supports outsourcing without the leakage of feature index, but it needs to pad the DAG to a complete tree; therefore, its communication size linearly depends on \( 2^d \); yet it has the best performance for small tree evaluations in the WAN setting when the network delay is 80ms. (cf. Sec. 8) With regards to polynomial-round solutions, [20] is the most efficient scheme in the literature; nevertheless, their offline communication depends on the tree size, and complete tree padding is needed to support outsourcing. Our polynomial-round scheme has logarithmic communication in both online and offline phase.

## 2 Preliminaries

**Notations.** Throughout this paper, we use the following notations and terminologies. Let \( \lambda \in \mathbb{Z} \) be the security parameter. Denote a value \( x \) indexed by a label \( b \) as \( x^{(b)} \), while \( x^{(b)} \) means the value of \( x \) power of \( b \). Denote a \((2, 2)\)-additive secret sharing in \( \mathbb{Z}_m \) by \( \llbracket x \rrbracket := \{x^{(1)}, x^{(2)}\} \), where \( x^{(1)} + x^{(2)} = x \mod n \). Denote a \((4, 4)\)-additive secret sharing in \( \mathbb{Z}_m \) by \( \llbracket x \rrbracket := \{x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}\} \), where \( x^{(1)} + x^{(2)} + x^{(3)} + x^{(4)} = x \mod n \). When \( K \) is a set, \( k \leftarrow K \) stands for sampling \( k \) uniformly at random from \( K \), and \( |K| \) stands for the size of \( K \) in terms of the number of elements. When \( f \) is an algorithm, \( y \leftarrow f(x) \) stands for running \( f \) on input \( x \). We map \( x \in [-2^{t-1}, 2^{t-1}] \) to \( \mathbb{Z}_{2^t} \), i.e., when \( x \) is negative, \( x' = x + 2^{t-1} \).

**Branching Program and Decision Tree.** In this work, we focus on the deterministic branching program based on DAG and support its generalizations to integer-valued sink labels and input features. Let \( B \) denote a branching program. \( B \) has a unique source node and one or more sink nodes. Each non-sink node of \( B \) corresponds to an input feature \( x \in \mathbb{Z}_{2^t} \) and has two outgoing edges labeled 0 or 1. Each sink node of \( B \) has a label \( v_i \in \mathbb{Z}_{2^t} \) that determines the output of \( B \) evaluation. For a \( B \), \( m \) is defined as the number of its nodes, \( m_1 \) is defined as the number of its non-sink nodes, and its depth \( d \) is the length of the longest path. A decision tree is a special branching program whose underlying DAG is a tree. Denote a decision tree by \( T \). Without loss of generality, we assume \( T \) is a binary tree, which can be met by converting a general tree to a binary tree. \( T \) follows the notations of \( B \). The leaves and root in \( T \) correspond to the sinks and source node in \( B \), respectively. In addition, each non-sink node of \( T \) has a comparison function for input feature \( x \in \mathbb{Z}_{2^t} \) and a given threshold \( t \in \mathbb{Z}_{2^t} \).

The evaluation of \( T \) or \( B \) is performed by traversing from the source node to a sink node. Thus the evaluation takes linear time with respect to \( d \). In detail, \( T \) or \( B \) receives an \( n \)-dimensional feature vector \( x := (x_i)_{i \in \mathbb{Z}_n} \) as evaluation input. Starting from the source node, for the \( i \)-th node, if current node is a non-sink node, fetch \( x_{k_i} \) from \( x \), where \( k_i \in \mathbb{Z}_n \) is the index of the corresponding feature. Then determine the next
When DPF is used to realize a PIR protocol, the servers need to run on every element of the input domain. Boyle et al. [5] provide a more efficient scheme for this case, rather than executing \(|G^m|\) independent invocations of Eval. We adopt their scheme and denote it by Eval\(\Delta\)(\(b, K^{(h)}\)).

Definition 1. Let \(T \subset [2]\). We say a two-party FSS scheme \((\text{Gen}, \text{Eval})\) is \(T\)-secure for function family \(\mathcal{F} = \{f : G^m \rightarrow G^{|\mathcal{E}|}\}\), if for all non-uniform PPT adversaries \(\mathcal{A}\), it holds that

\[
\text{Adv}(1^{|\mathcal{E}|}, \mathcal{A}) = \Pr \left[ (f_0, f_1, \phi) \leftarrow \mathcal{A}(1^{|\mathcal{E}|}); b \leftarrow \{0, 1\}; \right. \\
\left. (\mathcal{K}^{(1)}, \mathcal{K}^{(2)}) \leftarrow \text{Gen}(1^{|\mathcal{E}|}, f_b) ; \\
b^* \leftarrow \mathcal{A}(\mathcal{E})^i; \phi) : \\
f_0, f_1 \in \mathcal{F} \land b = b^* \right] - \frac{1}{2}
\]

is negligible in \(\lambda\).

3 System Architecture and Security Model

System architecture. Fig. 1 gives a high-level architecture of our outsourced private decision tree and BP evaluation platform. The entities consists of a set of four non-colluding computing servers \(S := \{S_1, \ldots, S_4\}\), the model owner \(M\), the data owner \(D\), and the receiver \(R\). Initially, the model owner shares its model \(M\) among the computing servers. For each evaluation, a subset of data owners provide their feature data to the computing servers in the shared form; the servers then obliviously evaluate the model on given data and output the result to a subset of the receivers.

Universal Composability. Our security model is based on the Universal Composibility (UC) framework [8], which lays down a solid foundation for designing and analyzing protocols secure against attacks in an arbitrary network execution environment (therefore it is also known as network aware security model). Roughly speaking, in the UC framework, protocols are carried out over multiple interconnected machines; to capture attacks, a network adversary \(\mathcal{A}\) is introduced, which is allowed to corrupt some machines (i.e., have the full control of all physical parts of some machines); in addition, \(\mathcal{A}\) is allowed to partially control the communication tapes of all uncorrupted machines, that is, it sees all the messages sent from and to the uncorrupted machines and controls the sequence in which they are delivered. Then, a protocol \(\rho\) is a UC-secure implementation of a functionality \(\mathcal{F}\), if it satisfies that for every network adversary \(\mathcal{A}\) attacking an execution of \(\rho\), there is another adversary \(\mathcal{S}\)—known as the simulator—attacking
It interacts with a set of computing servers $S := \{S_1, \ldots, S_k\}$, the model owner $M$, the data owner $D$, the receiver $R$, and the adversary $Sim$. It is parameterized with a set $F$ and a variable status.

Initially, set $F := \emptyset$ and status := 0.

**Outsourcing phase:**
- Upon receiving (MODEL, sid, $M$) from the model owner $M$:
  - Send notification (MODEL, sid, $M$, $(M.m, M.d)$) to $Sim$;
  - Set status := 1;
  - Record $M$;
- Upon receiving (DATA, sid, $x$) from the data owner $D$, if status = 1:
  - Send notification (DATA, sid, $D$, $|x|$) to $Sim$;
  - Set status := 2;
  - Record $x$;

**Evaluation phase:**
- Upon receiving (EVAL, sid) from server $S_i \in S$, if status = 2 does:
  - Send notification (Eval, sid, $S_i$) to $Sim$;
  - Set $F := F \cup \{S_i\}$;
  - If $|F| = k$, run $y \leftarrow M(x)$;
  - Send (RESULT, sid, $y$) to $R$ via input delayed channel;

**Functionality $F_{bp}^K$**

The ideal process that uses $F$ (by corrupting the same set of machines), such that, the executions of $p$ with $A$ and that of $F$ with $S$ makes no difference to any network execution environment.

**The idea world execution.** In the ideal world, the computing servers $S := \{S_1, \ldots, S_k\}$, the model owner $M$, the data owner $D$, and the receiver $R$ only communicate with an ideal functionality $F_{bp}^K$ during the evaluation. As depicted in Fig. 2, the ideal functionality $F_{bp}^K$ consists of two phases. In the outsourcing phase, the model owner $M$ sends its model $M$ to the ideal functionality. Later, the data owner $D$ sends its data $x$ to the ideal functionality. Note that the size and depth of the model as well as the number of features are leaked to the adversary $Sim$. During the evaluation phase, once all computing servers have sent (EVAL, sid) to the functionality $F_{bp}^K$, $F_{bp}^K$ runs $y \leftarrow M(x)$ and then sends (RESULT, sid, $y$) to $R$ via input delayed channel.

**Functionality $F_{sot}^N$**

It interacts with $S := \{S_1, \ldots, S_k\}$ and the adversary $Sim$.
- Upon receiving $(\text{FETCH}, \text{sid}, x(i)) := (x(0), \ldots, x(N-1), i)$ from $S_i \in S$:
  - Send notification (FETCH, sid, $S_j$) to $Sim$;
  - Record $(x(i), i)$;
- Once all players have submitted their input, does:
  - Assert $x(1) = x(2)$ and $x(3) = x(4)$;
  - Compute $i := \sum_{j=1}^4 i(j)$ (mod $N$);
- Upon receiving (RAND, sid, $y^*$) from $Sim$ for the corrupted party $S_i$:
  - Pick random $y(1), \ldots, y(4) \in \mathbb{Z}_{2^l}$ under the constraint $\sum_{j=1}^4 y(j) = x(1) + x(3)$ (mod $2^l$) and $y(k) = y^*$;
  - Send (RETURN, sid, $y(i)$) to all parties $S_j \in S$ via private delayed channel.

**Figure 3:** The shared OT functionality $F_{sot}^N$ simulator $Sim$ such that for all PPT environment $Z$ it holds:

$$\text{Exec}_{\Pi, A, Z} \approx \text{Exec}_{F_{bp}^K, Sim, Z}$$

## 4 OT with logarithmic communication

In the 1-out-of-$N$ shared OT protocol, given a vector of shared messages $x := (x_0, \ldots, x_{N-1})$ and an shared index $i \in \mathbb{Z}_N$, the MPC parties can jointly obtain $x_i$ in the shared form without revealing $i$. We propose an efficient shared OT protocol with logarithmic communication in both offline and online phase. As depicted in Fig. 3, our shared OT is a 4-party computation protocol. The messages are replicated shared, while the index is additively shared. Let $x(i) := (x(i)_0, \ldots, x(i)_{N-1})$ and $f(i)$ be the shares of player $S_i$, $j \in [4]$. We have $x^{(1)} = x^{(2)}$ and $x^{(3)} = x^{(4)}$; messages $x = x^{(1)} + x^{(3)} = x^{(2)} + x^{(4)}$ and $i = \sum_{j=1}^4 i(j)$ (mod $N$). To facilitate our private decision tree and BP evaluation protocol, the output of shared OT is additively shared among the four players; nevertheless, it is possible to add 1 round share conversion to any other shared type at the end.

**Intuition.** Our construction utilizes the DPF technique [5] in a novel way, and in this work the output of DPF for $f_i(x)$ is additive shared in $\mathbb{Z}_{2^l}$ instead of GF($2^l$). Conventionally, in a DPF-based two-server OT protocol, the client holds an index $i \in \mathbb{Z}_N$ and both servers hold the messages $x \in (\mathbb{Z}_{2^l})^N$. During the protocol, the client generates a pair of DPF keys $(\mathcal{K}^{(1)}, \mathcal{K}^{(2)})$ for $f_i(x)$ and then distributes them to the two servers. The servers then jointly evaluate and return shares of $x_i := \sum_{j=1}^N f_i(j) \cdot x_j$ to the client. On the other hand, in shared OT, the index and messages are both stored in the shared form. To address the former issue, we let a third server (an non-evaluator of this DPF), say $S_3$, generate a pair of
DPF keys \( (\mathcal{K}^{(1)}, \mathcal{K}^{(2)}) \) on \( f_{\Phi}(x) \) for a random \( \Phi \in \mathbb{Z}_N \) in the offline phase. \( S_3 \) then sends \( \mathcal{K}_1 \) and \( \mathcal{K}_2 \) to \( S_1 \) and \( S_2 \), respectively. In online phase, \( \delta := i - \Phi \mod N \) is opened to the evaluators, i.e., \( S_1 \) and \( S_2 \). The evaluators then cyclically shift the messages vector \( x \) to the left \( \delta \) positions and evaluate DPF \( f_{\Phi}(x) \) on the shifted messages to obtain \( x_j \). To address the latter issue, the messages are replicated shared, i.e., \( x := x^{(1)} \oplus x^{(2)} \oplus x^{(4)} \), such that \( S_1 \) and \( S_2 \) (\( S_3 \) and \( \delta \)) hold the same share; therefore, they can perform DPF evaluation on the shares instead of the plaintext.

**Protocol description.** Our 1-round shared OT protocol is designed in the online/offline model (cf. Fig. 5). During the initialization, \( S_1 \) and \( S_3 \) agree on a random seed \( \eta_1 \in \{0,1\}^k \); \( \delta_2 \) and \( \delta_4 \) agree on a random seed \( \eta_2 \in \{0,1\}^k \); \( S_1 \) and \( S_2 \) agree on a random seed \( \eta_3 \in \{0,1\}^k \); \( S_3 \) and \( S_4 \) agree on a random seed \( \eta_4 \in \{0,1\}^k \). In offline phase, \( S_1 \) and \( S_3 \) act as DPF generator locally invoke DPF \( f_{\Phi}(x) \). \( \Phi \) with random input \( \phi_1, \phi_2 \leftarrow \mathbb{Z}_N \) to get DPF keys \( \langle \mathcal{K}_{\phi_1}, \mathcal{K}_{\phi_2}\rangle \). In online phase, four servers compute \( \langle \delta_1 \rangle := \langle i \rangle - \langle x^{(1)} \rangle \) and \( \langle \delta_2 \rangle := \langle i \rangle - \langle x^{(2)} \rangle \) with fresh random mask \( w_1 \) and \( w_2 \). Then reveal \( \delta_1 \) to \( S_1 \) and \( S_2 \), \( \delta_2 \) to \( S_3 \) and \( S_4 \), as shown in Fig. 4b. For \( j \in \{1,2\} \), \( S_j \) first cyclic-shifts the share of messages \( x^{(j)} \) to the left \( \delta_j \) positions and denotes the array after shift as \( \tilde{x}^{(j)} \). Next, \( S_j \) invokes DPF \( f_{\Phi_1}(x) \). \( \Phi_1 \) with the DPF key that received in offline phase. After that, \( S_1 \) and \( S_2 \) obtain a secret shared array \( \langle \tilde{\beta}_{k,\phi_1}, k \in \mathbb{Z}_N \rangle \) whose the only non-zero value is \( \tilde{\beta}_{\phi_1,\phi_1} = 1 \). We have \( x_j = \sum_{j=1}^{N-1} \tilde{y}^{(j)} \) (mod 2^\ell) where

\[
\tilde{y}^{(j)} := \sum_{k=0}^{N-1} (x_k^{(j)} \cdot \tilde{\beta}_{k,\phi_1}) \mod 2^\ell.
\]

Finally, we re-randomize \( \tilde{y}^{(j)} \) to ensure the uniform distribution.

**Efficiency.** \( \Pi_{\text{ot}}^{N,\ell} \) is a one-round 1-out-of-\( N \) shared OT protocol with offline communication cost \( 4\lambda \log N \) bits and online communication cost \( 12\ell \) bits.

**Security.** We show the security of our 1-out-of-\( N \) shared OT Protocol \( \Pi_{\text{ot}}^{N,\ell} \) with the following theorem, and its proof can be found in Appendix A.
It interacts with $S := \{S_1, \ldots, S_\ell\}$ and the adversary $Sim$.

- Upon receiving $(\text{COMP}F\text{ETCH}, \text{sid}, x^{(j)}, m^{(j)})$ from $S_j \in S$:
  - Send notification $(\text{COMP}F\text{ETCH}, \text{sid}, S_j)$ to $Sim$;
  - Record $(x^{(j)}, m^{(j)})$;
- Once all players have submitted their input, do:
  - For $k \in [0, 1]$, compute $x_t := \sum_{j=1}^{\ell} x_{t,j}^{(j)}$ (mod $2^\ell$) and $m_{t} := \sum_{j=1}^{\ell} m_{t,j}^{(j)}$ (mod $2^\ell$);
  - Set $b \leftarrow (m_0 < m_1)$;
- Upon receiving $(\text{RAND}, \text{sid}, y^*)$ from $Sim$ for the corrupted party $S_i$:
  - Pick random $y^{(i)}, \ldots, y^{(\ell)} \in \mathbb{Z}_{2^\ell}$, under the constraint
    $\sum_{j=1}^{\ell} y^{(j)} = x_t - b$ (mod $2^\ell$) and $y^{(j)} = y^*$;
  - Send $(\text{RETURN}, \text{sid}, y^{(j)})$ to all parties $S_j \in S$ via private delayed channel.

Figure 6: The conditional shared OT functionality $\mathcal{F}^{(i_1,i_2)}_{\text{csot}}$.

**Theorem 1.** Let $\text{DPF}_{\mathbb{Z}_2^\ell, \mathbb{Z}_2^\ell}$ be a secure function secret sharing scheme for point function $f_{\alpha, \beta}(x) : \mathbb{Z}_2^\ell \rightarrow \mathbb{Z}_2^\ell$ with adversarial advantage $\text{Adv}_{\text{DPF}}^{\mathbb{Z}_2^\ell, \mathbb{Z}_2^\ell}(1^\lambda, \mathcal{A})$. Let $\mathcal{F}^{\mathbb{Z}_2^\ell, \mathbb{Z}_2^\ell}(0, 1)^\lambda \times \{0, 1\}^\infty \rightarrow \mathbb{Z}_2^\ell$ be a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}}^{\mathbb{Z}_2^\ell, \mathcal{A}}(1^\lambda)$. The protocol $\Pi_{\text{csot}}'$ as described in Fig. 5 UC-realizes $\mathcal{F}^{\mathbb{Z}_2^\ell, \mathbb{Z}_2^\ell}_{\text{csot}}$ as described in Fig. 3 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage

$$3 \cdot \text{Adv}_{\text{PRF}}^{\mathbb{Z}_2^\ell, \mathcal{A}}(1^\lambda) + \text{Adv}_{\text{DPF}}^{\mathbb{Z}_2^\ell, \mathbb{Z}_2^\ell}(1^\lambda, \mathcal{A}) \; .$$

5 Conditional Shared OT

In the conditional shared OT protocol, given a vector of shared messages $x := (x_0, x_1) \in (\mathbb{Z}_2^\ell)^2$ and two shared keywords $m := (m_0, m_1) \in (\mathbb{Z}_2^\ell)^2$, the MPC players first securely compare $b \leftarrow (m_0 < m_1)$ and then obtain $x_1 - b$ in the shared form without revealing $b$. As depicted in Fig. 6, our conditional shared OT is a 4-party computation protocol. The messages and keywords are additively shared among the 4 parties. Let $x^{(j)} := (x_0^{(j)}, x_1^{(j)})$ and $m^{(j)} := (m_0^{(j)}, m_1^{(j)})$ be the shares of player $S_j, j \in [4]$. We have $x = \sum_{i=1}^{\ell} x^{(i)}$ and $m = \sum_{i=1}^{\ell} m^{(i)}$.

**Intuition.** Naively, the conditional shared OT protocol can be realized by a secure comparison followed by an oblivious selection (a.k.a. multiplication) protocol. However, this would result a 2-round protocol. We compress the round complexity to one. In our protocol, the servers are divided into two groups. $\Delta m := m_1 - m_0$ is opened to each group with the corresponding DCF offset, while the $(4,4)$-additive secret sharing messages are converted to replicated secret sharing, where the servers of each group have the same shares. The two groups then perform two DCF evaluations in a parallel. Subsequently, the oblivious selection can be computed locally.

Figure 7: Conditional shared OT Protocol $\Pi_{\text{csot}}^{(i_1,i_2)}$.
Figure 8: Communication diagram for protocol \(\Pi^{f_1,f_2}_{\text{cot}}\).

by scalar product.

**Protocol description.** Our 1-round conditional shared OT is depicted in Fig. 7. During the initialization, \(S_1\) and \(S_3\) agree on a random seed \(\eta_1 \in \{0,1 \}^\lambda\); \(S_2\) and \(S_4\) agree on a random seed \(\eta_2 \in \{0,1 \}^\lambda\); \(S_1\) and \(S_2\) agree on a random seed \(\eta_3 \in \{0,1 \}^\lambda\); \(S_3\) and \(S_4\) agree on a random seed \(\eta_4 \in \{0,1 \}^\lambda\).

In offline phase, \(S_1\) invokes \(\text{DCF}^{2,2}_{\text{f1}}\).Gen\(^\text{IC}\) with random offset \(\rho_2\) and input \(2^\ell - 1\) (the decomposition point of positive and negative numbers) to generate \((x_1^{(1)}, x_1^{(2)})\), then distributes them to \(S_1\) and \(S_2\); \(S_3\) invokes \(\text{DCF}^{2,2}_{\text{f1}}\).Gen\(^\text{IC}\) with the other random offset \(\rho_1\) and input \(2^\ell - 1\) to generate \((x_2^{(1)}, x_2^{(2)})\), then distributes them to \(S_1\) and \(S_2\).

In online phase, four servers reveal \(\langle \Delta m_1 \rangle := \langle m_1 \rangle - \langle m_1 \rangle + \rho_1\) to \(S_1\) and \(S_2\); reveal \(\langle \Delta m_2 \rangle := \langle m_1 \rangle - \langle m_2 \rangle + \rho_2\) to \(S_3\) and \(S_4\). Meanwhile, \(S_1\) and \(S_4\) generate \(\hat{x}^{(3)} := (\hat{x}_0^{(3)}, \hat{x}_1^{(3)})\) and \(\hat{x}^{(4)} := (\hat{x}_0^{(4)}, \hat{x}_1^{(4)})\) by \(\text{PRF}\) with the same random seed \(\eta_4\) respectively, such that \(\hat{x}^{(3)} = \hat{x}^{(4)}\); then four servers compute \(\langle \hat{x} \rangle := \langle \hat{x} \rangle \pmod{2^\ell}\) and reveal it to \(S_1\) and \(S_2\) with the help of fresh random mask \(w_1\) and \(w_2\). \(S_1\) denotes \(\hat{x}\) by \(\hat{x}_1^{(1)}\), \(\hat{x}_2^{(2)}\) denotes \(\hat{x}\) by \(\hat{x}_2^{(2)}\), where messages \(x = \hat{x}^{(1)} + \hat{x}^{(3)} = \hat{x}^{(2)} + \hat{x}^{(4)}\) as shown in Fig. 8. After that, servers locally evaluate \(\text{DCF}\) with received key and masked input to obtain the shared comparison result \(\beta_1, \beta_2\), then compute scalar multiplication and re-randomize the result to get the shares of selected message \((y)\) in uniform distribution.

**Efficiency.** \(\Pi^{f_1,f_2}_{\text{cot}}\) is a one-round protocol with offline communication cost \(4\lambda \log \ell_2\) bits and online communication cost \( 12(\ell_1 + \ell_2)\) bits.

**Security.** We show the security of our conditional shared OT Protocol \(\Pi^{f_1,f_2}_{\text{cot}}\) with the following theorem, and its proof can be found in Appendix B.

**Theorem 2.** Let \(\text{DCF}^{2,2}_{\text{f1}}\) be a secure function secret sharing scheme for offset comparison function \(f^{\text{IC}}_{\text{f1}}(x)\): \(\mathbb{Z}_{2^2} \rightarrow \mathbb{Z}_{2^1}\) with adversarial advantage \(\text{Adv}_{\text{DCF}^{2,2}_{\text{f1}}}(1^\lambda, \mathcal{A})\). Let \(\text{PRF}^{2,1}_{\text{f1}} : \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \mathbb{Z}_{2^1}\) be a secure pseudorandom function with adversarial advantage \(\text{Adv}_{\text{PRF}^{2,1}_{\text{f1}}}(1^\lambda, \mathcal{A})\). Let \(\text{PRF}^{2,2}_{\text{f2}} : \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \mathbb{Z}_{2^2}\) be a secure pseudorandom function with adversarial advantage \(\text{Adv}_{\text{PRF}^{2,2}_{\text{f2}}}(1^\lambda, \mathcal{A})\). The protocol \(\Pi^{f_1,f_2}_{\text{cot}}\) as described in Fig. 7 UC-realizes \(\mathcal{F}^{f_1,f_2}_{\text{cot}}\) as described in Fig. 6 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage

\[
8 \cdot \text{Adv}_{\text{PRF}^{2,1}_{\text{f1}}}(1^\lambda, \mathcal{A}) + 3 \cdot \text{Adv}_{\text{PRF}^{2,2}_{\text{f2}}}(1^\lambda, \mathcal{A}) + \text{Adv}_{\text{DCF}^{2,2}_{\text{f1}}}(1^\lambda, \mathcal{A}).
\]

6 Private Decision Tree and BP Evaluation

In this section, we propose two solutions for outsourced private decision tree and BP evaluation. The first solution is a constant-round protocol for (small) complete trees; whereas, the second solution is a polynomial-round protocol for BP and (large) sparse tree evaluation.

6.1 Constant-Round Protocol

Our constant-round protocol requires three communication rounds and a complete decision tree, which can be trans-
Outsourcing Protocol $\Pi_{os}^{\text{const}}$

- Upon receiving $(\text{MODEL}, \text{sid}, (\mathcal{P}, \psi))$ from the environment $\mathcal{Z}$, the model owner $M$:
  
  - foreach element $i$ in $\mathcal{P}$:
    
    - Set $k_i^{(1)} \leftarrow Z_n$, $k_i^{(3)} \leftarrow k_i - k_i^{(1)} \pmod{n}$
    
    - Set $t_i^{(1)} \leftarrow Z_{2^2}$, $t_i^{(2)} \leftarrow t_i^{(1)} \pmod{2^2}$
    
    - Set $P_i := (k_i^{(1)}, t_i^{(1)}), P_i^{(2)} := (k_i^{(3)}, t_i^{(2)})$
  
- foreach element $j$ in $\psi$:
  
  - $v_j^{(1)} \leftarrow Z_{2^2}$, $v_j^{(2)} \leftarrow v_j - v_j^{(1)} \pmod{2^2}$
  
  - Send $(P_i, v_i^{(1)})$ to $S_1$, $(P_i^{(2)}, x_i^{(2)})$ to $S_2$
  
- Upon receiving $(\text{DATA}, \text{sid}, \mathbf{x})$ from the environment $\mathcal{Z}$, the data owner $D$:
  
  - for each $i$:
    
    - Generate $x_i^{(1)} \leftarrow Z_{2^2}$, set $x_i^{(2)} := x_i^{(1)}$
    
    - Set $x_i^{(3)} := x_i - x_i^{(1)} \pmod{2^2}$, $x_i^{(4)} := x_i^{(3)}$
    
    - Send $x_i^{(2)}$ to $S_j$, $j \in [4]$

Figure 11: Outsourcing Protocol $\Pi_{os}^{\text{const}}$.

formed from a normal DAG by adding dummy nodes as illustrated in Fig. 9, i.e. $\bar{m} = 2^d$, $\bar{m}_i = 2^{d-1} - 1$ . All leaf nodes extended by dummy decision nodes have the same classification value as real path. We use a vector, denoted as $\mathcal{P}$, to represent all decision nodes and complete tree structure. Each $P_i \in \mathcal{P}$ consists of the input selection index $k_i$ and a threshold value $t_i$. The left and right child of $P_i$ are $P_{2i+1}$ and $P_{2i+2}$, respectively. The leaf nodes’ classification values form the other vector, denoted as $\psi$.

Our protocol selects corresponding features and compares thresholds with them for all decision nodes. For each $P_i \in \mathcal{P}$, $S_1$ and $S_2$ obliquely set its “selected” out-going edge cost (based on the comparison result) to 0, and set the other outgoing edge cost to random value. Then $S_1$ and $S_2$ sum up the share of edge costs along all paths to get a vector of path costs for all leaf nodes in a shared form. As shown in Fig 10, only one path cost takes the value of 0 and the corresponding leaf nodes’ classification value is the evaluation result.

Outsourcing. First of all, the model owner $M$ invokes $\Pi_{os}^{\text{const}}$ as described in Fig. 11 to generate the additive share of $\mathcal{P}, \psi$ and distribute them to $S_1$ and $S_2$. This step only needs to be performed once for a given model. Before the start of each evaluation, the data owner $D$ shares the input features $\mathbf{x} := (x_i)_{i \in \mathbb{Z}_n}$ to four servers in replicated secret sharing. After the execution, for $j \in [4], S_j$ hold the shares $x_j^{(i)} := (x_i^{(j)})_{i \in \mathbb{Z}_n}$, such that $\mathbf{x}^{(1)} = \mathbf{x}^{(2)}, \mathbf{x}^{(3)} = \mathbf{x}^{(4)}$ and $\mathbf{x} = \mathbf{x}^{(1)} + \mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \mathbf{x}^{(4)}$.

Evaluation. Our constant-round protocol follows the modular design framework of [18]. As depicted in Fig. 12, it consists of feature selection, comparison and path evaluation.

Feature selection. For each $P_i \in \mathcal{P}$, with the secret shared index $[k_i]$ in $S_1$ and $S_2$, four servers construct the $(4,4)$-secret-sharing $(\hat{k}_i)$ by setting $k_i^{(1)} := 0$ and $k_i^{(3)} := 0$ in $S_1$ and $S_4$. Then four servers invoke our 1-out-of-N shared OT

Figure 12: Constant-round Evaluation Protocol $\Pi_{eval}^{\text{const}}$ in the $\mathcal{F}_{\text{sort}}$-hybrid model.
Our comparison protocol is based on the DCF scheme [4], where $S_4$ plays the role of generator while $S_1$ and $S_2$ play the role of evaluator. To avoid leaking features and thresholds to servers, we let $S_4$ precompute the DCF keys for all $P_i \in \mathcal{P}$, which compares the input value with a random value $p_i$ and distribute the keys to evaluators $S_1$ and $S_2$. Then $S_1$ and $S_2$ are able to obtain shared comparison result vector $\{b_i\}_{i \in \mathbb{Z}_{m+1}}$, where $b_i = 1$ if $t_i - x_i$ is positive and $b_i = 0$ otherwise.

**Path evaluation.** $S_1$ and $S_2$ first generate random masks $(r_i)_{i \in \mathbb{Z}_{m+1}}$ together. Next, for each decision node $P_i \in \mathcal{P}$, $S_1$ and $S_2$ locally compute its left out-going edge cost $\|e_{iL}\| := \|(1-b_i) \cdot r_i\|$ and right out-going edge cost $\|e_{iR}\| := \|b_i \cdot r_i\|$. For each leaf node $v_i \in \mathcal{V}$, $S_1$ and $S_2$ sum up the edge costs along its path from the root to get its path cost $c_i$ in shared form. Subsequently, $S_1$ and $S_2$ generate a random number $\delta \leftarrow \mathbb{Z}_{m+1}$ together. Denote path cost vector as $C$ (cf. Fig. 10). $S_1$ and $S_2$ circular shift the $\{C_i\}$ and $\{v_i\}$ to the left $\delta$ positions. Then open $C$ and reshare $v$ to $S_1$, $S_2$, $S_3$ and $S_4$ can easily select the classification value share $v_i$ of evaluation result according to the position $i$ of $c_i = 0$. Finally, they return the $v_i^{(1)}$ and $v_i^{(2)}$ to the receiver.

## 6.2 Polynomial-Round Protocol

**Security.** We show the security of our constant-round protocol ($\Pi_{\text{const}}^{\text{sec}}$, $\Pi_{\text{eval}}^{\text{const}}$) with the following theorem, and its proof can be found in Appendix C.

**Theorem 3.** Let $\text{DPF}_{\mathcal{IC}}^{\mathbb{Z}_m, \mathbb{Z}_2^\lambda}$ be a secure function secret sharing scheme for offset comparison function $f_{\alpha, \beta}(x) : \mathbb{Z}_m \mapsto \mathbb{Z}_2^\lambda$ with adversarial advantage $\text{Adv}_{\text{DCF}_{\mathcal{IC}}^{\mathbb{Z}_m, \mathbb{Z}_2^\lambda}}(1^\lambda, \mathcal{A})$. Let $\text{PRF}_{\mathbb{Z}_m, \mathbb{Z}_2^\lambda}^{\mathbb{Z}_m, \mathbb{Z}_2^\lambda} : \{0,1\}^\lambda \times \{0,1\}^m \mapsto \mathbb{Z}_2^\lambda$ be a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}_{\mathbb{Z}_m, \mathbb{Z}_2^\lambda}}^{\mathbb{Z}_m, \mathbb{Z}_2^\lambda}(1^\lambda, \mathcal{A})$. The protocol $\Pi_{\text{const}}^{\text{sec}}$ as described in Fig. 11 and $\Pi_{\text{eval}}^{\text{const}}$ as described in Fig. 12 UC-realizes $\mathcal{F}_d^{\text{dp}}$ as described in Fig. 2 in the $\mathcal{F}_\text{IC}$ hybrid model against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage

$$3m_c \cdot \text{Adv}_{\text{PRF}_{\mathbb{Z}_m, \mathbb{Z}_2^\lambda}}^{\mathbb{Z}_m, \mathbb{Z}_2^\lambda}(1^\lambda, \mathcal{A}) + (m_c + 1) \cdot \text{Adv}_{\text{DCF}_{\mathcal{IC}}^{\mathbb{Z}_m, \mathbb{Z}_2^\lambda}}(1^\lambda, \mathcal{A})$$

Our polynomial-round protocol supports sparse tree and BP evaluation. To hide the model structure, we introduce only one dummy node instead of transforming the sparse decision tree into a full tree, i.e. $\tilde{m} = m + 1$. Let the dummy node point to itself and all leaf nodes point to it as shown in Fig. 13. The main idea is that, during privacy-preserving evaluation, once a sink node is reached, servers will obliviously access this dummy node (repeatedly) until the protocol reaches $d$ steps. Thus, the length of evaluation path is always $d$.

We use a vector to describe this padded model, which includes all kinds of nodes. Without confusion, we also denote it as $\mathcal{P}$. Each $P_i \in \mathcal{P}$ consists of the index $I_i^{\text{left}}$ and the input selection index $I_i^{\text{right}}$ of its left child, the index $I_i^{\text{left}}$ and the input selection index $I_i^{\text{right}}$ of its right child, a threshold value $t_i$ and a classification value $v_i$ of $P_i$. If $P_i$ represents the dummy node, $I_i^{\text{left}}$ and $I_i^{\text{right}}$ take the value of the index of dummy node $\tilde{m}$, $J_i^{\text{left}}$ and $J_i^{\text{right}}$ take random values, and $v_i$ is equal to 0. If $P_i$ represents a decision node, $v_i$ is dummy data such that $v_i = 0$. If $P_i$ represents a sink node, $I_i^{\text{left}}$, $I_i^{\text{right}}$, $J_i^{\text{left}}$ and $J_i^{\text{right}}$ are the same dummy data as the dummy node. Since there only is one leaf node in a path, and only if $v$ belongs to a leaf node the value of $v$ is non-zero, the accumulation of
• Upon receiving \((\text{MODEL}, \id, \mathcal{P})\) from the environment \(\mathcal{Z}\), the model owner \(\mathcal{M}\):
  - Build the position mapping, denote i-th element as \(P_i := \{I_{\text{left}}(i), I_{\text{right}}(i)\} \subseteq \mathbb{Z}_m\);
  - for \(i := 0 \to m-1\) do:
    * Set \(I_{\text{left}}(0) := I_{\text{left}}(1)\);
    * Set \(I_{\text{left}}(1) := I_{\text{left}}(2)\) \(\Rightarrow \) \(I_{\text{left}}(3) := I_{\text{left}}(4)\) \(\mod \bar{m}\);
    * Set \(I_{\text{left}}(1) := I_{\text{right}}(1)\) \(\Rightarrow \) \(I_{\text{left}}(3) := I_{\text{right}}(3)\) \(\mod \bar{m}\);
    * Set \(I_{\text{left}}(3) := I_{\text{right}}(4)\) \(\Rightarrow \) \(I_{\text{left}}(1) \mod n\);
    * Set \(I_{\text{left}}(1) := Z_{\bar{m}}, I_{\text{right}}(1) := I_{\text{right}}(2)\);
    * Set \(I_{\text{left}}(2) := I_{\text{right}}(3)\) \(\Rightarrow \) \(I_{\text{left}}(3) := I_{\text{right}}(4)\);
    * Set \(I_{\text{left}}(3) := Z_{\bar{m}}, I_{\text{right}}(3) := I_{\text{right}}(2)\);
    * Set \(I_{\text{left}}(4) := I_{\text{right}}(1)\) \(\Rightarrow \) \(I_{\text{left}}(2) := I_{\text{right}}(3)\);
    * Set \(I_{\text{left}}(2) := Z_{\bar{m}}, I_{\text{right}}(2) := I_{\text{right}}(1)\);
    * Set \(t_1 := \bar{m}t_1(t_1) \mod 2^3\);
    * Set \(v := \bar{m}v \mod 2^3\);
    - Set \(i \in \{1, 2, 3\} \Rightarrow \) \(v_i := v - v_i \mod 2^3\);
  - Set \(v_1, v_2, v_3 \in \mathbb{Z}_2\);
  - Set \(x_0 := x_0 + v_1\), \(x_1 := x_1 + v_2\), \(x_2 := x_2 + v_3\);
  - Send \(x_i\) to \(S_{i+1}\), \(i \in \{1, 2, 3\}\).

Figure 15: Outsourcing Protocol \(\Pi_{\text{os}}^{\text{poly}}\).

\(v\) of all the nodes in the evaluation path is exactly equal to the classification value of the reached leaf node.

Our polynomial-round protocol requires \(2d\) rounds. Referring to the example in Fig. 14, for \(i\)-th step in the evaluation, servers first obliviously fetch the “current node” \(P_i\) and the appropriate feature \(x_{i_k}\) in the Round 1. Then compute:

\[
\begin{align*}
\text{res} := \text{res} + v_i, \\
\quad c &\leftarrow (x_{i_k} < t_i).
\end{align*}
\]

and indicates the next node index is \(I_{\text{left}}(c = 1)\) or \(I_{\text{right}}(c = 0)\) in the Round 2. After repeating the above process \(d\) times, the \(\langle \text{res} \rangle\) is open to receiver as the evaluation result.

Outsourcing. For polynomial-round protocol, the data owner outsourcing protocol is identical to our constant-round scheme, but the model owner outsourcing protocol is different. As described in Fig. 15, for each \(P_i \in \mathcal{P}\), the model owner \(M\) generates replicated shares \(P_i(j), j \in [4]\). Send them to \(S_1, S_2, S_3, S_4\) respectively. In order to make servers aware of the evaluation entry, \(M\) shares the element index \(i_d\) and the feature selection index \(k_{i_d}\) of the root node to four servers in \(\langle 4, 4 \rangle\)-additive secret sharing.

Theorem 4. The protocol \(\Pi_{\text{os}}^{\text{poly}}\) as described in Fig. 15 and \(\Pi_{\text{eval}}^{\text{poly}}\) as described in Fig. 16 UC-realizes \(\mathcal{F}_{\text{os}}\) and \(\mathcal{F}_{\text{eval}}\) as described in Fig. 2 in the \(\langle \mathcal{F}_{\text{os}}, \mathcal{F}_{\text{eval}} \rangle\)-hybrid model against semi-honest adversaries who can statically corrupted up to 1 server.
Table 2: Parameters of the models in the UCI dataset.

<table>
<thead>
<tr>
<th>Decision Tree</th>
<th>Features</th>
<th>Depth</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Wine</td>
<td>7</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>Linnerud</td>
<td>3</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>Breast</td>
<td>12</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>Digits</td>
<td>47</td>
<td>15</td>
<td>168</td>
</tr>
<tr>
<td>Spambase</td>
<td>57</td>
<td>17</td>
<td>58</td>
</tr>
<tr>
<td>Diabetes</td>
<td>10</td>
<td>28</td>
<td>393</td>
</tr>
<tr>
<td>Boston</td>
<td>13</td>
<td>30</td>
<td>425</td>
</tr>
</tbody>
</table>

7 Malicious Security

We can upgrade the proposed protocols to tolerate malicious adversary; that is the malicious behavior can be detected but without identifiable abort. First of all, as shown in [4], it is possible to modify the DPF and DCF schemes such that the correctness of the evaluation result can be tested when one of the evaluators is malicious. More specifically, we generate two pairs of FSS keys. One pair express the same point function (or less than function) \( f \) as in semi-honest protocol, and the other express \( f_{\alpha} := \alpha f \), where \( \alpha \leftrightarrow G_{\text{out}} \). To detect malicious FSS key generator behavior, we let the two non-evaluation servers (say \( S_1 \) and \( S_2 \)) generate the same key pairs with a common random seed; therefore, the corresponding keys are identical. We let them independently send the FSS keys\(^1\) to the evaluation servers \( S_3 \) and \( S_4 \), respectively. If one of the FSS key generators is malicious, the FSS keys would be different. We rest MPC operations can be upgraded to achieve malicious security using the linear MAC techniques presented in SPDZ protocols [9, 10].

8 Implementation and Benchmarks

The proposed constant-round scheme and polynomial-round scheme are implemented in C++. The DCF and polynomial-round schemes are improved from [23]. Since Ma et al. [20] did not release their source code, we re-implement their scheme using AES-NI and EMP-toolkits [25]. In addition, the state-of-the-art constant-round protocols are adopted from [17] for performance comparison. Our benchmarks are executed on a desktop with Intel(R) Core i7 8700 CPU @ 3.2 GHz running Ubuntu 18.04.2 LTS; with 6 CPUs, 32 GB Memory and 1TB SSD. There network environments are simulated: local-area network (LAN, RTT: 0.1ms, bandwidth: 1Gbps), metropolitan-area network (MAN, RTT: 6ms, bandwidth: 100Mbps), and wide-area network (WAN, RTT: 80ms, bandwidth: 40Mbps).

Our experiment uses datasets from the UCI machine learning repository [13], which consists of Iris, Wine (chemical analysis), Linnerud (physical exercise performance), Breast (cancer), Digits, Spambase, Diabetes, and Boston (housing value). Their concrete parameters are shown in Table 2.2. We set secure parameter \( \lambda \) to 128, feature bit-length \( \ell \) to 64. Note that the performance results of the related works, e.g., MTZC [20], are slightly different from that presented in the original papers due to different implementation and experiment environment. The main overhead of the offline phase of our protocol is to generate the FSS key. Table 3 shows the offline phase performance comparison between our polynomial-round protocol and MTZC [20] in the outsourced setting. Compared with MTZC [20], our protocol is slightly slower for small DAG models, while it is about 4X faster for big DAG models.

Fig.17 illustrates the online runtime comparison between our two protocols and the related works. The results are taken as the average of 10 evaluations. We fail to obtain the evaluation results for Diabetes and Boston models for our constant-round protocol and MTZC outsourcing protocols, as both protocols require complete-tree padding. For depth \( d = 28, 30 \) trees, complete decision tree padding would cause the memory out of computer capacity.

In a network environment with higher bandwidth and lower latency such as the LAN setting, our polynomial-round protocol runs much more faster than the state-of-the-arts. More precisely, our polynomial-round protocol is up to 15X faster than the others in the LAN setting. In a network environment with lower bandwidth and higher latency such as the WAN setting, our constant-round protocol outperforms the state-of-the-art protocols. In particular, our constant-round protocol is up to 10X faster than the others in the WAN setting.

9 Related Work

There has been a huge literature in private BP and/or decision tree evaluation. The first work is proposed by Ishai and Paskin [15]. They evaluate a BP on encrypted input via homomorphic public-key cryptosystem, and require \( O(md) \) communication. It is impractical for cases with a large number of input features, like medical diagnosis. And their protocol does not include comparison in each non-sink node.

Later, many evaluation protocols are proposed also with constant communication round. Brikkell et al. [7] present a private diagnosis system based on BP model. They imple-
Figure 17: Online runtime (in different log scales) in LAN/MAN/WAN (bandwidth/RTT) setting with Intel(R) Core i7 8700 CPU @ 3.2 GHz running Ubuntu 18.04.2 LTS, 32 GB Memory and 1TB SSD. Ours(Poly)/Ours(Const) refers to our polynomial-round/constant-round protocol. MTZC [20] refers to the sparse tree variant; MTZC-Outsourcing refers to their outsourcing variant.

On the other hand, constant-round protocols above always require the client to have at least linear computation in the model size \(m\), which is not friendly to weak client with limited computational resource. Thus, researchers are attracted to pursue new solutions with sublinear computation complexity for client, i.e., the parties can only adaptively perform necessary feature selections and comparisons along with the evaluation path. The main idea is to obliviously select only one decision node for comparison at each layer of the DAG via either OT or ORAM, such as [16] and [22]. The dependence of the current selection on previous comparison results leads to the round complexity of protocol is usually linear in the length \(d\) of the longest path.

Recently, the outsourcing extension is considered in private feature selection with additive HE (AHE) and oblivious transfer (OT), and transform the whole BP into a secure program consisting of GCs representing permuted nodes to evaluate comparisons. [3] treats a decision tree as a high-degree polynomial with a priori fixed multiplicative depth and evaluate the polynomial through costly full HE (FHE) to obtain result. [26] gets rid of FHE by using DGK protocol based on AHE for comparison and OT for leaf node selection. But [26] requires a complete tree (with dummy nodes) and permuting it. [21] improves [26] by a new “path cost” approach, which is a linear function for each path and determines whether a leaf node contains the classification result. Their protocol is purely based on AHE, without introducing dummy nodes. Obviously, [26] and [21] take advantage of the properties of the tree structure, thus no longer support BP evaluation. [18] systematically reviews prior constant-round solutions and proposes a modular construction from three constant-round sub-protocols: (a) private feature selection, (b) secure comparison, and (c) oblivious path evaluation. [18] also identifies novel combinations of these linear sub-protocols that provide better tradeoffs.

On the other hand, constant-round protocols above always require the client to have at least linear computation in the model size \(m\), which is not friendly to weak client with limited computational resource. Thus, researchers are attracted to pursue new solutions with sublinear computation complexity for client, i.e., the parties can only adaptively perform necessary feature selections and comparisons along with the evaluation path. The main idea is to obliviously select only one decision node for comparison at each layer of the DAG via either OT or ORAM, such as [16] and [22]. The dependence of the current selection on previous comparison results leads to the round complexity of protocol is usually linear in the length \(d\) of the longest path.

Recently, the outsourcing extension is considered in private feature selection with additive HE (AHE) and oblivious transfer (OT), and transform the whole BP into a secure program consisting of GCs representing permuted nodes to evaluate comparisons. [3] treats a decision tree as a high-degree polynomial with a priori fixed multiplicative depth and evaluate the polynomial through costly full HE (FHE) to obtain result. [26] gets rid of FHE by using DGK protocol based on AHE for comparison and OT for leaf node selection. But [26] requires a complete tree (with dummy nodes) and permuting it. [21] improves [26] by a new “path cost” approach, which is a linear function for each path and determines whether a leaf node contains the classification result. Their protocol is purely based on AHE, without introducing dummy nodes. Obviously, [26] and [21] take advantage of the properties of the tree structure, thus no longer support BP evaluation. [18] systematically reviews prior constant-round solutions and proposes a modular construction from three constant-round sub-protocols: (a) private feature selection, (b) secure comparison, and (c) oblivious path evaluation. [18] also identifies novel combinations of these linear sub-protocols that provide better tradeoffs.

10 Concluding Remarks

We presented a 4-server MPC platform for outsourced private decision tree and BP evaluation. For uniformity, we assume each BP decision node also has a comparison; however, it can be easily removed to adapt to any other binary decision diagram. Our key building block is a lightweight 1-out-of-\(N\) shared OT protocol with logarithmic communication. Unlike [12], we utilize the DPF scheme in a novel way such that the ORAM functionality is achieved without the need of oblivious PRF evaluation via MPC. Our polynomial-round outsourced private decision tree evaluation protocol achieves

\[O(d^2)\] communication. In addition, none of the above outsourced evaluation protocol support privacy-preserving BP evaluation.

---

2We consider the secure outsourcing without the leakage of the index mapping between decision nodes and input features. [19] and [1] do not meet this condition.
logarithmic communication in both online and offline; Yet, it is unknown if there exists a constant-round protocol with logarithmic overall communication. We leave this as an open problem. As a future work, we will extend our 4-server MPC platform to support RAM programs.

References


A Proof of Theorem 1

Theorem 1. Let $DPF_{\mathcal{Z}^N,\mathcal{Z}^N'}$ be a secure function secret sharing scheme for point function $f_{a,b}(x) : \mathcal{Z}^N \rightarrow \mathcal{Z}^N'$ with adversarial advantage $Adv_{DPF_{\mathcal{Z}^N,\mathcal{Z}^N'}}(1^\lambda, \mathcal{A})$. Let $PRF_{\mathcal{Z}^N} : \{0,1\}^\lambda \times \{0,1\}^m \rightarrow \mathcal{Z}^N$ be a secure pseudorandom function with adversarial advantage $Adv_{PRF_{\mathcal{Z}^N}}(1^\lambda, \mathcal{A})$. The protocol $\Pi_{\mathcal{Z}^N}$ as described in Fig. 5 UC-realizes $\mathcal{F}_{\mathcal{Z}^N}^{N,\ell}$ as described in Fig. 3 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage

$$3 \cdot Adv_{PRF_{\mathcal{Z}^N}}(1^\lambda, \mathcal{A}) + Adv_{DPF_{\mathcal{Z}^N}}(1^\lambda, \mathcal{A})$$

Proof. To prove Thm. 1, we construct a PPT simulator $Sim$ such that no uniform PPT environment $\mathcal{Z}$ can distinguish between (i) the real execution $\text{Exec}_{\mathcal{Z}^N,\mathcal{Z}^N,\mathcal{A},\mathcal{Z}}$ where the parties $S := \{S_1, \ldots, S_4\}$ run protocol $\Pi_{\mathcal{Z}^N}$ in the real world and the corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$, and (ii) the ideal execution $\text{Exec}_{\mathcal{Z}^N,\mathcal{A},\mathcal{Z}}$ where the parties $S_1, \ldots, S_4$ interact with functionality $\mathcal{F}_{\mathcal{Z}^N,\mathcal{A},\mathcal{Z}}$ in the ideal world, and corrupted parties are controlled by the simulator $Sim$. We consider following cases.

Case 1: $S_1$ (or $S_2$) is corrupted.

Simulator. The simulator $Sim$ internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Sim simulates the interface of honest parties $S_2, S_3, S_4$. In addition, the simulator $Sim$ simulates the following interactions with $\mathcal{A}$.

- Upon initialization, the simulator $Sim$ acts as the honest party $S_3$ to do:
  - Generate $\phi_1, \phi_1^{(2)} \leftarrow \mathcal{Z}_N$ and set $\phi_1^{(1)} := \phi_1 - \phi_1^{(2)}$;
  - Set $\mathcal{K}_\phi^{(1)}, \mathcal{K}_\phi^{(2)} \leftarrow DPF_{\mathcal{Z}_N, \mathcal{Z}_N'; (1^\lambda, \phi_1, 1, \mathcal{Z}_N, \mathcal{Z}_N')}$;
  - Send $\mathcal{K}_\phi^{(1)}$ to $S_1, \mathcal{K}_\phi^{(2)}$ to $S_2$;
  - The simulator $Sim$ does:
    - Pick random $w_1^{(1)}, w_1^{(2)} \leftarrow \mathcal{Z}_N$;
    - Set $w_1^{(3)} := w_1^{(1)} - w_1^{(2)}$;
  - Upon receiving $(\text{FETCH}, \text{sid}, S_j)$ for an honest party $S_j, j \in \{1,2\}$ from the external $\mathcal{F}_{\mathcal{Z}^N,\ell}$, the simulator $Sim$ does:
    - Set $\delta_1^{(j)} := w_1^{(j)}$ and $\delta_2^{(j)} := 0$;
    - Send $\delta_1^{(j)}$ to $S_3$ and $\delta_2^{(j)}$ to $S_3$ on behalf of $S_j$;
  - Upon receiving $(\text{FETCH}, \text{sid}, S_j)$ for an honest party $S_j, j \in \{3,4\}$ from the external $\mathcal{F}_{\mathcal{Z}^N,\ell}$, the simulator $Sim$ does:
    - Set $\delta_1^{(j)} := -\phi_1^{(j-2)} - w_1^{(j)}$, and $\delta_2^{(j)} := 0$;
    - Send $\delta_1^{(j)}$ to $S_1$ and $S_2$, $\delta_2^{(j)}$ to $S_{7-j}$ on behalf of $S_j$;
  - Upon receiving $\delta_1^{(1)}$ from the corrupted $S_1$ to $S_2$ and $\delta_2^{(1)}$ from the corrupted $S_1$ to $S_3, S_4$, the simulator $Sim$ does:
    - Extract $\ell^{(1)} := \delta_1^{(1)} - PRF_{\eta_1}(\text{sid}, 1) \mod N$;
    - Send $(\text{FETCH}, \text{sid}, \mathbf{x}^{(2)}, \ell^{(1)})$ to the external $\mathcal{F}_{\mathcal{Z}^N,\ell}$;
    - Compute $\delta_1 := \delta_1^{(1)} + \delta_1^{(2)} + \delta_1^{(3)} + \delta_1^{(4)} \mod N$;
Claim 1. If PRF
\textsuperscript{Z,N}: \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \mathbb{Z}_N is a secure pseudorandom function with adversarial advantage Adv\textsubscript{PRF\textsuperscript{Z,N}}(1^\lambda, \mathcal{A})$, then \mathcal{H} \textsubscript{2} and \mathcal{H} \textsubscript{0} are indistinguishable with advantage \epsilon \textsubscript{2} := 3 \cdot Adv\textsubscript{PRF\textsuperscript{Z,N}}(1^\lambda, \mathcal{A})

Proof. We have changed 3 PRF outputs to uniformly random strings; therefore, the overall advantage is 3 \cdot Adv\textsubscript{PRF\textsuperscript{Z,N}}(1^\lambda, \mathcal{A}) by hybrid argument via reduction.

It is straightforward that the distribution of \{\delta_1^{(j)}\}_{j \in [4]} are uniformly random under the condition \delta_1 := \sum_{k=1}^{4} \delta_1^{(k)} = i - \varphi_1, where \varphi_1 is used to generate the DPF keys \varphi_0^{(1)}, \varphi_0^{(2)} \rightarrow DPF.Gen(1^\lambda, \varphi_0, 1, \mathbb{Z}_N, \mathbb{Z}_2^\lambda). Whereas \delta_1 := -\varphi_1 in the hybrid \mathcal{H} \textsubscript{2}, we can show that if there exists an adversary \mathcal{A} who can distinguish the view of \mathcal{H} \textsubscript{2} from the view of \mathcal{H} \textsubscript{0} then we can construct an adversary \mathcal{B} who uses \mathcal{A} in a blackbox fashion to break DPF.Eval\textsuperscript{Z,N,2}\textsuperscript{f} := (Gen, Eval) with the same advantage. Therefore, \mathcal{H} \textsubscript{2} and \mathcal{H} \textsubscript{0} are indistinguishable with adversarial advantage \epsilon \textsubscript{2} := Adv\textsubscript{DPF.Eval\textsuperscript{Z,N,2}\textsuperscript{f}}(1^\lambda, \mathcal{A})

The adversary’s view of \mathcal{H} \textsubscript{2} is identical to the simulated view Exec\textsubscript{DPF.Eval\textsuperscript{Z,N,2}\textsuperscript{f}}\textsubscript{\mathcal{S}, \mathcal{Z}}, i.e. the overall distinguishing advantage is

\[ 3 \cdot Adv\textsubscript{PRF\textsuperscript{Z,N}}(1^\lambda, \mathcal{A}) + Adv\textsubscript{DPF.Eval\textsuperscript{Z,N,2}\textsuperscript{f}}(1^\lambda, \mathcal{A}) \]

Case 2: S_3 (or S_4) is corrupted.

Simulator. The simulator Sim internally runs \mathcal{A}, forwarding messages to/from the environment \mathcal{Z}. Sim simulates the interface of honest parties S_1, S_2, S_4. In addition, the simulator Sim simulates the following interactions with \mathcal{A}.
- Upon initialization, the simulator Sim acts as the honest party S_1 to do:
  - Generate \varphi_0, \varphi_2 \leftarrow \mathbb{Z}_N and set \varphi_1^{(1)} := \varphi_0 - \varphi_2^{(2)};
  - Set \varphi_0^{(1)}, \varphi_0^{(2)} \rightarrow DPF.Gen(1^\lambda, \varphi_0, 1, \mathbb{Z}_N, \mathbb{Z}_2^\lambda);
  - Send \varphi_0^{(1)} to S_3, \varphi_0^{(2)} to S_4;
- The simulator Sim does:
  - Pick random \omega_1^{(1)}, \omega_1^{(2)} \leftarrow \mathbb{Z}_N;
  - Set \omega_1^{(3)} := w_1^{(1)}, \omega_1^{(4)} := w_1^{(2)};
- Upon receiving (FETCH, sid, S_j) for an honest party S_j, j \in \{1,2\} from the external \mathcal{F}_{\mathcal{S}, \mathcal{Z}}, the simulator Sim does:
  - Set \delta_1^{(j)} := 0 and \delta_1^{(j)} := \omega_1^{(2)} - \varphi_1^{(j)};
  - Send \delta_1^{(j)} to S_{3-j}, \delta_1^{(j)} to S_{3-j} on behalf of S_j;
- Upon receiving (FETCH, sid, S_j) for an honest party S_j, j \in \{3,4\} from the external \mathcal{F}_{\mathcal{S}, \mathcal{Z}}, the simulator Sim does:
  - Set \delta_1^{(j)} := 0 and \delta_1^{(j)} := -w_1^{(j)};
  - Send \delta_1^{(j)} to S_1 and \delta_1^{(j)} to S_{7-j} on behalf of S_j;
- Upon receiving \delta_1^{(j)} from the corrupted S_j to S_1, S_2 and \delta_1^{(j)} from the corrupted S_j to S_4, the simulator Sim does:
  - Extract \delta_1^{(j)} := \delta_1^{(3)} + PRF_{\eta_1, \text{sid}}(1) (mod N);
  - Send (FETCH, sid, x^{(4)}, \delta_1^{(3)}) to the external \mathcal{F}_{\mathcal{S}, \mathcal{Z}};
  - Compute \delta_2 := \delta_1^{(2)} + \delta_2^{(2)} + \delta_3^{(3)} (mod N);
  - Set \delta_2^{(3)} := x^{(4)} (mod N), for k \in \mathbb{Z}_N;
  - Set (\beta_{k, \varphi_2}^{(1)}) \leftarrow DPF.EvalAll(1, \varphi_0^{(1)});
  - Set \zeta_1 \leftarrow PRF_{\varphi_0^{(1)}}(\text{sid}), \zeta_4 \leftarrow PRF_{\varphi_0^{(2)}}(\text{sid});
  - Compute \gamma^{(3)} := \sum_{k=0}^{N-1} (x_0^{(1)} \cdot B_{k, \varphi_0^{(2)}} - \zeta_1 - \zeta_4 \mod 2^6);
  - Send (RAND, sid, \gamma^{(3)}) to the external \mathcal{F}_{\mathcal{S}, \mathcal{Z}};
Indistinguishability. We assume that the parties $S_1, \ldots, S_4$ communicate with each other via the secure channel functionality $f_{sc}$ (omitted in the protocol description for simplicity). The indistinguishability is proven through a series of hybrid worlds $\mathcal{H}_0, \ldots, \mathcal{H}_3$.

**Hybrid $\mathcal{H}_0$:** It is the real protocol execution $\text{Exec}_{\rho^{N_{1/2}}, \mathcal{A}, Z'}$.

**Hybrid $\mathcal{H}_1$:** $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, $\{w_j^{(j)}\}_{j \in [2]}$ and $\varphi_2^{(j)}$ are picked uniformly random from $\mathbb{Z}_N$ instead of calculating from $\text{PRF}_N$. Set $w_2^{(3)} := w_2^{(1)}, w_2^{(4)} := w_2^{(2)}$.

**Claim 3.** If $\text{PRF}^{N_{1/2}} : \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \mathbb{Z}_N$ is a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}^{N_{1/2}}} \left(1^\lambda, \mathcal{A}\right)$, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\varepsilon_1 := 3 \cdot \text{Adv}_{\text{PRF}^{N_{1/2}}} \left(1^\lambda, \mathcal{A}\right)$.

**Proof.** We have changed 3 PRF outputs to uniformly random strings; therefore, the overall advantage is $3 \cdot \text{Adv}_{\text{PRF}^{N_{1/2}}} \left(1^\lambda, \mathcal{A}\right)$ by hybrid argument via reduction.

**Hybrid $\mathcal{H}_2$:** $\mathcal{H}_2$ is the same as $\mathcal{H}_1$ except that in $\mathcal{H}_2$:
- For $j \in \{1, 2\}$, set $\delta_1^{(j)} := 0$ and $\delta_2^{(j)} := w_2^{(j)} - \varphi_2^{(j)}$.
- For $j \in \{3, 4\}$, set $\delta_1^{(j)} := 0$ and $\delta_2^{(j)} := -w_2^{(j)}$.

   Instead of:
- For $j \in \{1, 2\}$:
  - Set $\delta_1^{(j)} := i^{(j)} + w_1^{(j)}$ and $\delta_2^{(j)} := i^{(j)} - \varphi_2^{(j)} + w_2^{(j)}$.
- For $j \in \{3, 4\}$:
  - Set $\delta_1^{(j)} := i^{(j)} - \varphi_1^{(j)} - w_1^{(j)}$ and $\delta_2^{(j)} := i^{(j)} - w_2^{(j)}$.

**Claim 4.** If $\text{DPF}^{Z_N,Z_{4/3}} := (\text{Gen, Eval})$ is a secure function secret sharing scheme for point function $f_{\alpha, \beta}(x) : \mathbb{Z}_N \rightarrow \mathbb{Z}_{4/3}$ with adversarial advantage $\text{Adv}_{\text{DPF}}^{Z_N,Z_{4/3}} \left(1^\lambda, \mathcal{A}\right)$, then $\mathcal{H}_3$ and $\mathcal{H}_2$ are indistinguishable with advantage $\varepsilon_2 := \text{Adv}_{\text{DPF}}^{Z_N,Z_{4/3}} \left(1^\lambda, \mathcal{A}\right)$.

**Proof.** Note that the corrupted party $S_1$ only sees $\{\delta_1^{(j)}\}_{j \in [4]}$: therefore, the modification of $\{\delta_1^{(j)}\}_{j \in [4]}$ is oblivious to $S_3$. In the hybrid $\mathcal{H}_1$, we have
- $\delta_2^{(1)} := i^{(1)} - \varphi_2^{(1)} + w_2^{(1)}$;
- $\delta_2^{(2)} := i^{(2)} - \varphi_2^{(2)} + w_2^{(2)}$;
- $\delta_2^{(3)} := i^{(3)} - w_2^{(3)}$;
- $\delta_2^{(4)} := i^{(4)} - w_2^{(4)}$.

It is straightforward that the distribution of $\{\delta_1^{(j)}\}_{j \in [4]}$ are uniformly random under the condition $\delta_2 := \sum_{k=1}^{4} \delta_2^{(k)} = i - \varphi_2$, where $\varphi_2$ is used to generate the DPF keys $\varphi_1^{(1)}, \varphi_2^{(2)} \leftarrow \text{DPF.Gen}(1^\lambda, \varphi_2, 1, \mathbb{Z}_N, \mathbb{Z}_{4/3})$. Whereas $\delta_2 := -\varphi_2$ in the hybrid $\mathcal{H}_2$, we can show that if there exists an adversary $\mathcal{A}$ who can distinguish the view of $\mathcal{H}_2$ from the view of $\mathcal{H}_3$ then we can construct an adversary $\mathcal{B}$ who uses $\mathcal{A}$ in a blackbox fashion can break $\text{DPF}^{Z_N,Z_{4/3}} := (\text{Gen, Eval})$ with the same advantage. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_3$ are indistinguishable with adversarial advantage $\varepsilon_2 := \text{Adv}_{\text{DPF}}^{Z_N,Z_{4/3}} \left(1^\lambda, \mathcal{A}\right)$.

The adversary’s view of $\mathcal{H}_2$ is identical to the simulated view $\text{Exec}_{\rho^{N_{1/2}}, \mathcal{A}, Z'}$. Therefore, the overall distinguishing advantage is

$$3 \cdot \text{Adv}_{\text{PRF}^{N_{1/2}}} \left(1^\lambda, \mathcal{A}\right) + \text{Adv}_{\text{DPF}}^{Z_N,Z_{4/3}} \left(1^\lambda, \mathcal{A}\right).$$

This concludes the proof.

**B Proof of Theorem 2**

**Theorem 2.** Let $\text{DCF}_{1/2} : \mathbb{Z}_N \rightarrow \mathbb{Z}_{1/2}$ be a secure function secret sharing scheme for offset comparison function $f_{\alpha, \beta}(x) : \mathbb{Z}_N \rightarrow \mathbb{Z}_{1/2}$ with adversarial advantage $\text{Adv}_{\text{DCF}_{1/2}} \left(1^\lambda, \mathcal{A}\right)$. Let $\text{PRF}^{Z_N} : \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \mathbb{Z}_{1/2}$ be a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}^{Z_N}} \left(1^\lambda, \mathcal{A}\right)$. Let $\text{PRF}^{Z_{1}} : \{0,1\}^\lambda \times \{0,1\}^m \rightarrow \mathbb{Z}_{1/2}$ be a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}^{Z_{1}}} \left(1^\lambda, \mathcal{A}\right)$. The protocol $\Pi_{1/2} \mathcal{F}_{1/2}$ as described in Fig. 7 UC-realizes $\mathcal{F}_{1/2}$ as described in Fig. 6 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage

$$8 \cdot \text{Adv}_{\text{PRF}^{Z_N}} \left(1^\lambda, \mathcal{A}\right) + 3 \cdot \text{Adv}_{\text{PRF}^{Z_{1}}} \left(1^\lambda, \mathcal{A}\right) + \text{Adv}_{\text{DCF}_{1/2}} \left(1^\lambda, \mathcal{A}\right).$$

**Proof.** To prove Thm. 2, we construct a PPT simulator $\text{Sim}$ such that no non-uniform PPT environment $\mathcal{Z}$ can distinguish between (i) the real execution $\text{Exec}_{\Pi_{1/2} \mathcal{F}_{1/2}, \mathcal{A}, Z}$ where the parties $S := \{S_1, \ldots, S_4\}$ run protocol $\Pi_{1/2} \mathcal{F}_{1/2}$ in the real world and the corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$, and (ii) the ideal execution $\text{Exec}_{\Pi_{1/2} \mathcal{F}_{1/2}, \mathcal{Sim}, Z}$ where the parties $S_1, \ldots, S_4$ interact with functionality $\mathcal{F}_{1/2}$ in the ideal world, and corrupted parties are controlled by the simulator $\text{Sim}$. We consider following cases.

**Case 1:** $S_1$ (or $S_2$) is corrupted.

**Simulator.** The simulator $\text{Sim}$ internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. $\text{Sim}$ simulates the interface of honest parties $S_2, S_3, S_4$. In addition, the simulator $\text{Sim}$ simulates the following interactions with $\mathcal{A}$.

- Upon initialization, the simulator $\text{Sim}$ acts as the honest party $S_1$ to do:
  - Generate $\rho_1, \rho_2 \leftarrow \mathbb{Z}_{1/2}$, and set $\rho_1^{(1)} := \rho_1 - \rho_2$;
  - $\varphi_1^{(1)}, \varphi_2^{(2)} \leftarrow \text{DCF.Gen}(1^\lambda, 2^{1/2 - 1}, 1, \mathbb{Z}_{1/2}, \mathbb{Z}_{1/2}, \rho_1)$;
  - Send $\varphi_1^{(1)}$ to $S_1, \varphi_2^{(2)}$ to $S_2$;
- The simulator $\text{Sim}$ does:
  - Pick random $w_1^{(1)}, w_2^{(2)}, w_3^{(1)}, w_4^{(1)}, w_5^{(1)}$.
• Upon receiving \( \text{COMP\text{F}etch,\text{sid},S_j} \) for an honest party \( S_j \), \( j \in \{1,2\} \) from the external \( \mathcal{F}_c^{N,\ell} \), Sim does:
  - Set \( \Delta m_{p_1} := w_{m,1}, \Delta m_{p_2} := 0; \)
  - Set \( x_{i,j} := w_{x,i}, \) for \( i \in \{0,1\}; \)
  - Send \( \{\Delta m_{p_1}, \hat{x}_{i,j}\} \) to \( S_3,S_4 \) on behalf of the honest party \( S_j \);
• Upon receiving \( \text{COMP\text{F}etch,\text{sid},S_j} \) for an honest party \( S_j \), \( j \in \{3,4\} \) from the external \( \mathcal{F}_c^{N,\ell} \), Sim does:
  - Set \( \Delta x_{i,j} := r_1 + r_2 \mod 2^{f(i)}, \hat{x}_{i,j} := r_3 + r_4 \mod 2^{f(i)}; \)
  - Set \( \Delta m_{p_1} := \rho_1^{(j-2)} - w_{m,1} \mod 2^{f(i)}, \Delta m_{p_2} := 0; \)
  - Set \( \Delta x_{i,j} := -r_1 + r_2 - w_{x,2} \mod 2^{f(i)}, i \in \{0,1\}; \)
  - Send \( \{\Delta m_{p_1}, \hat{x}_{i,j}\} \) to \( S_3 \) and \( S_2, \Delta m_{p_2} \) to \( S_7;j \);
• Upon receiving \( \{\Delta m_{p_1}, \hat{x}_{i,j}\} \) from the corrupted \( S_1 \) to \( S_2 \) and \( \Delta m_{p_2} \) from the corrupted \( S_1 \) to \( S_3,S_4 \), the simulator Sim does:
  - For \( i \in \{0,1\} \), extract \( x_{i,j} := x_{i,j} - w_{x,i} \mod 2^{f(i)}; \)
  - Extract \( \Delta m_{p_1} := \rho_{1}^{(j-2)} + w_{m,1} \mod 2^{f(i)}; \)
  - Pick random \( m_0^{(1)}, m_1^{(1)} \leftarrow \mathbb{Z}_2 \) s.t. \( m_1^{(1)} - m_0^{(1)} = \Delta m_{p_1} \); \)
  - Send \( \{\Delta m_{p_1}, \hat{x}_{i,j}\} \) to the external \( \mathcal{F}_c^{c_{f,i};z}; \)
  - Set \( \Delta m_{p_1} := \sum_{i=1}^{4} \Delta m_{p_1} \mod 2^{f(i)}; \)
  - Set \( \hat{x}_{i,j} := \sum_{i=1}^{4} \hat{x}_{i,j} \mod 2^{f(i)}; \)
  - Set \( \hat{x}_{i,j} := \sum_{i=1}^{4} \hat{x}_{i,j} \mod 2^{f(i)}; \)
  - Set \( \hat{p}_{1}^{(1)} := \text{DCF, Eval}_{\bar{A}}^{z}, \Delta m_{p_1}; \)
  - Set \( \tilde{\xi}_i := \text{PRF}_{\bar{A}}^{z},(\text{sid},3), \tilde{\zeta}_i := \text{PRF}_{\bar{A}}^{z},(\text{sid},1); \)
  - Compute \( y_{i,j} := \tilde{p}_{1}^{(1)} \cdot y_{i,j} - x_{i,j} \mod 2^{f(i)}; \)
  - Send \( \{\text{RAND},y_{i,j}\} \) to the external \( \mathcal{F}_c^{c_{f,i};z}; \)

Indistinguishability. We assume that the parties \( S_1, \ldots, S_4 \) communicate with each other via the secure channel functionality \( \mathcal{F}_c \) (omitted in the protocol description for simplicity). The indistinguishability is proven through a series of hybrid worlds \( \mathcal{H}_0, \ldots, \mathcal{H}_6 \).

**Hybrid \( \mathcal{H}_0 \):** It is the real protocol execution \( \mathcal{F}_c^{c_{f,i};z}. \)

**Hybrid \( \mathcal{H}_6 \):** \( \mathcal{H}_6 \) is the same as \( \mathcal{H}_0 \) except that in \( \mathcal{H}_1 \), \( \{w_{x,0}\}_{i\in[2]}, \{w_{x,1}\}_{i\in[2]} \) and \( \{r_{0}\}_{j\in[4]} \) are picked uniformly random from \( \mathbb{Z}_2^{f(1)} \) instead of calculating from \( \text{PRF}_{\bar{A}}^{z}. \)

**Claim 5.** If \( \text{PRF}_{\bar{A}}^{z} : \{0,1\}^k \times \{0,1\}^m \rightarrow \mathbb{Z}_2^{f(1)} \) is a secure pseudorandom function with adversarial advantage \( \text{Adv}_{\text{PRF}_{\bar{A}}^{z}}(1^{\lambda},\mathcal{A}) \), \( \text{Adv}_{\text{PRF}_{\bar{A}}^{z}}(1^{\lambda},\mathcal{A}) \) is a secure pseudorandom function with adversarial advantage \( \text{Adv} \) and \( \mathcal{H}_6 \) and \( \mathcal{H}_0 \) are indistinguishable with advantage \( \varepsilon_1 := 8 \cdot \text{Adv} + 3 \cdot \text{Adv} \).

**Proof.** We have changed 8 \( \text{PRF}_{\bar{A}}^{z} \) outputs and 3 \( \text{PRF}_{\bar{A}}^{z} \) outputs to uniformly random strings; therefore, the overall advantage is \( 8 \cdot \text{Adv} + 3 \cdot \text{Adv} \) by hybrid argument via reduction.

**Hybrid \( \mathcal{H}_2 \):** \( \mathcal{H}_2 \) is the same as \( \mathcal{H}_0 \) except that in \( \mathcal{H}_6 \):
  - For \( j \in \{1,2\} \):
    - Set \( \Delta m_{p_1} := w_{m,1} \) and \( \Delta m_{p_2} := 0; \)
  - For \( j \in \{3,4\} \):
    - Set \( \Delta m_{p_1} := \rho_1^{(j-2)} - w_{m,1} \mod 2^{f(i)}; \)
  - Instead of:
    - For \( j \in \{1,2\} \):
      - Set \( \Delta m_{p_1} := m_1^{(1)} - m_0^{(1)} + w_{m,1} \mod 2^{f(i)}; \)
      - Set \( \Delta m_{p_2} := m_1^{(1)} - m_0^{(1)} + \rho_2^{(j-2)} - w_{m,1} \mod 2^{f(i)}; \)
    - For \( j \in \{3,4\} \):
      - Set \( \Delta m_{p_1} := m_1^{(1)} - m_0^{(1)} + \rho_1^{(j-2)} - w_{m,1} \mod 2^{f(i)}; \)
      - Set \( \Delta m_{p_2} := m_1^{(1)} - m_0^{(1)} - w_{m,2} \mod 2^{f(i)}; \)

**Claim 6.** If \( \text{DCF, Eval}_{\bar{A}}^{z} : \{\text{Gen}_{\bar{A}}^{z}, \text{Eval}_{\bar{A}}^{z}\} \) is a secure function secret sharing scheme for offset comparison function \( \mathcal{F}_{\alpha,\beta}^{z}(1^{\lambda},\mathcal{A}) \), \( \mathcal{H}_2 \) and \( \mathcal{H}_1 \) are indistinguishable with advantage \( \varepsilon_2 := \text{Adv} \).

**Proof.** Note that the corrupted party \( S_1 \) only sees \( \{\Delta m_{p_2}^{(j)}\}_{j \in[4]} \); therefore, the modification of \( \{\Delta m_{p_1}^{(j)}\}_{j \in[4]} \) is oblivious to \( S_1 \). In the hybrid \( \mathcal{H}_6 \), we have:
  - \( \Delta m_{p_1} := m_1^{(1)} - m_0^{(1)} + w_{m,1} \mod 2^{f(i)}; \)
  - \( \Delta m_{p_2} := m_1^{(2)} - m_0^{(2)} + w_{m,2} \mod 2^{f(i)}; \)
  - \( \Delta m_{p_1} := m_1^{(3)} - m_0^{(3)} + \rho_1^{(1)} - \tilde{m}_{3,1} \mod 2^{f(i)}; \)
  - \( \Delta m_{p_1} := m_1^{(4)} - m_0^{(4)} + \rho_2^{(1)} - \tilde{m}_{4,1} \mod 2^{f(i)}; \)

It is straightforward that the distribution of \( \{\Delta m_{p_1}^{(j)}\}_{j \in[4]} \) are uniformly random under the condition \( \Delta m_{p_1} := \sum_{j=1}^{4} \Delta m_{p_1}^{(j)} = m_1 - m_0 + \rho_1 + \rho_2 \), where \( \rho_1 \), \( \rho_2 \) are used to generate the DCF keys \( \rho_1^{(1)}, \rho_1^{(2)} \) and \( \text{DCF, Gen}_{\bar{A}}^{z}(1^{\lambda}, \mathbb{Z}_2^{f(1)}, \mathbb{Z}_2^{f(2)}, \rho_1) \).
Hybrid $\mathcal{H}_2$: $\mathcal{H}_2$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_2$:

- For $j \in \{1, 2\}$:
  - Set $x_{j}^{(i)} := w_{x, i}^{(j)} \pmod{2^f_1}$, $i \in \{0, 1\}$;
- For $j \in \{3, 4\}$:
  - Set $x_{j}^{(i)} := -r_{j} + 2i - 2 - w_{x, i}^{(j)} \pmod{2^f_1}$, $i \in \{0, 1\}$; instead of
- For $j \in \{1, 2\}$:
  - Set $x_{j}^{(i)} := x_{j}^{(i)} + w_{x, i}^{(j)} \pmod{2^f_1}$, for $i \in \{0, 1\}$;
- For $j \in \{3, 4\}$:
  - Set $x_{j}^{(i)} := x_{j}^{(i)} - r_{j} + 2i - 2 - w_{x, i}^{(j)} \pmod{2^f_1}$, $i \in \{0, 1\}$;

**Claim 7.** $\mathcal{H}_3$ and $\mathcal{H}_2$ are perfectly indistinguishable.

**Proof.** Since $\{r_{j} \} \in [4]$ and $\{w_{x, i}^{(j)} \}$ are uniformly random in $\mathbb{Z}_{2^f_1}$, the distribution of $\{x_{j}^{(i)} \}_{j \in [4]}$ and $\{x_{j}^{(i)} \}_{j \in [4]}$ are identical. Therefore, $\mathcal{H}_3$ and $\mathcal{H}_2$ are perfectly indistinguishable.

The adversary’s view of $\mathcal{H}_6$ is identical to the simulated view $\mathcal{E}_{\text{ext}}^{\frac{f_1}{2}}_{\mathcal{S}, Z}$. Therefore, the overall distinguishing advantage is

$$8 \cdot \text{Adv}_{\text{PRF}}^{\frac{f_1}{2}}(1^{\lambda}, \mathcal{A}) + 3 \cdot \text{Adv}_{\text{PRF}}^{\frac{f_2}{2}}(1^{\lambda}, \mathcal{A}) + \text{Adv}_{\text{DFI}}^{\frac{f_1}{2}}(1^{\lambda}, \mathcal{A}).$$

**Case 2:** $S_3$ (or $S_4$) is corrupted.

**Simulator.** The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $Z$. Sim simulates the interface of honest parties $S_1, S_2, S_3$. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$:

- Upon initialization, the simulator Sim acts as the honest party $S_1$ to do:
  - Generate $\rho_2, \rho_2^{(2)} \leftarrow \mathcal{Z}_{2^f_2}$ and set $\rho_2^{(1)} := \rho_2 - \rho_2^{(2)}$;
  - $\mathcal{A}^{(2)} \leftarrow \text{DFC.Gen}(1^{\lambda}, 2^f_2 - 1, 1, \mathbb{Z}_{2^f_2}, \mathbb{Z}_{2^f_2}, \rho_2)$;
  - Send $\mathcal{A}^{(1)}$ to $S_1, \mathcal{A}^{(2)}$ to $S_2$;
- The simulator Sim does:
  - Pick random $w_{x, 0}^{(1)}, w_{x, 0}^{(2)}, w_{x, 1}^{(1)}, w_{x, 1}^{(2)}, r_{1}, r_{2}, r_{3}, r_{4} \leftarrow \mathcal{Z}_{2^f_2}$;
  - Pick random $w_{m, 0}^{(2)}, w_{m, 1}^{(2)} \leftarrow \mathcal{Z}_{2^f_2}$;
  - Set $w_{x, 0}^{(3)} := w_{x, 0}^{(1)}, w_{x, 0}^{(4)} := w_{x, 0}^{(2)}, w_{x, 1}^{(3)} := w_{x, 1}^{(1)}, w_{x, 1}^{(4)} := w_{x, 1}^{(2)}, w_{m, 2}^{(3)} := w_{m, 2}^{(1)}, w_{m, 2}^{(4)} := w_{m, 2}^{(2)}$;
- Upon receiving $(\text{COMP.Fetch}, \text{sid}, S_j)$ for an honest party $S_j, j \in \{1, 2\}$ from the external $\mathcal{E}_{\text{ext}}^{\frac{f_1}{2}}$, Sim does:
  - Set $\Delta m_{p_1}^{(j)} := 0$, $\Delta m_{p_2}^{(j)} := \rho_2^{(j)} + w_{m, 2}^{(j)} \pmod{2^f_2}$;
  - Set $x_{j}^{(i)} := w_{x, i}^{(j)}$, for $i \in \{0, 1\}$;
  - Send $(\Delta m_{p_1}^{(j)}, \hat{x}_{j}^{(j)})$ to $S_{3-j}, \Delta m_{p_1}^{(j)}$ to $S_3$ and $S_4$ on behalf of the honest party $S_j$;
- Upon receiving $(\text{COMP.Fetch}, \text{sid}, S_j)$ for an honest party $S_j, j \in \{3, 4\}$ from the external $\mathcal{E}_{\text{ext}}^{\frac{f_1}{2}}$, Sim does:
  - Set $x_{j}^{(i)} := r_1 + r_2 \pmod{2^f_1}$, $x_{j}^{(i)} := r_3 + r_4 \pmod{2^f_1}$;
  - Set $\Delta m_{p_1}^{(j)} := 0$, $\Delta m_{p_2}^{(j)} := -w_{m, 2}^{(j)} \pmod{2^f_2}$;
  - Set $x_{j}^{(i)} := -r_{j} + 2i - 2 - w_{x, i}^{(j)} \pmod{2^f_1}$, $i \in \{0, 1\}$;
  - Send $(\Delta m_{p_1}^{(j)}, \hat{x}_{j}^{(j)})$ to $S_1$ and $S_2, \Delta m_{p_2}^{(j)}$ to $S_{7-j}$;

**Claim 8.** If $\text{PRF}^{Z_{2^f_1}} : \{0, 1\}^{\lambda} \times \{0, 1\}^{19} \mapsto Z_{2^f_1}$ is a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}}^{Z_{2^f_1}}(1^{\lambda}, \mathcal{A})$, $\text{PRF}^{Z_{2^f_2}} : \{0, 1\}^{\lambda} \times \{0, 1\}^{19} \mapsto Z_{2^f_2}$ is a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}}^{Z_{2^f_2}}(1^{\lambda}, \mathcal{A})$, then $\mathcal{H}_4$ and $\mathcal{H}_5$ are indistinguishable with advantage $\epsilon := 8 \cdot \text{Adv}_{\text{PRF}}^{Z_{2^f_1}}(1^{\lambda}, \mathcal{A}) + 3 \cdot \text{Adv}_{\text{PRF}}^{Z_{2^f_2}}(1^{\lambda}, \mathcal{A})$.

**Proof.** We have changed 8 PRF$^{Z_{2^f_1}}$ outputs and 3 PRF$^{Z_{2^f_2}}$ outputs to uniformly random strings; therefore, the overall advantage is $8 \cdot \text{Adv}_{\text{PRF}}^{Z_{2^f_1}}(1^{\lambda}, \mathcal{A}) + 3 \cdot \text{Adv}_{\text{PRF}}^{Z_{2^f_2}}(1^{\lambda}, \mathcal{A})$ by hybrid argument via reduction.

**Hybrid $\mathcal{H}_6$:** $\mathcal{H}_6$ is the same as $\mathcal{H}_1$ except that in $\mathcal{H}_6$:

- For $j \in \{1, 2\}$:
  - Set $\Delta m_{p_1}^{(j)} := 0$ and $\Delta m_{p_2}^{(j)} := \rho_2^{(j)} + w_{m, 2}^{(j)} \pmod{2^f_2}$;
- For $j \in \{3, 4\}$:
Let $A$ be oblivious to $\{\rho_j\}$.

Note that the corrupted party $-\Delta - \Delta - \Delta - \Delta$

$m \cdot m \cdot m \cdot m$.

$\rho_j(\rho_j(3)) = 2$.

Therefore, the overall distinguishing advantage

Adv$_{\text{DCF}^\text{IC}}$($1^\lambda, A$).

Claim 9. If $\text{DCF}_\text{IC}^{Z_{2^f}, Z_{2^f}, t_1} := (\text{Gen}^\text{IC}, \text{Eval}^\text{IC})$ is a secure function secret sharing scheme for offset comparison function $f_{\alpha, \beta}(\lambda) : Z_{2^f} \rightarrow Z_{2^f}$, with adversarial advantage

Adv$_{\text{DCF}^\text{IC}}^{Z_{2^f}, Z_{2^f}, t_1}(1^\lambda, A)$, then $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\varepsilon_2 := \text{Adv}_{\text{DCF}^\text{IC}}^{Z_{2^f}, Z_{2^f}, t_1}(1^\lambda, A)$.

Proof. Note that the corrupted party $\mathcal{S}_3$ only sees $\{\Delta_m^{(j)}\}_{j \in [4]}$: therefore, the modification of $\{\Delta_m^{(j)}\}_{j \in [4]}$ is oblivious to $\mathcal{S}_1$. In the hybrid $\mathcal{H}_2$, we have

- $\Delta_m^{(1)} := m_1 - m_0 + \rho_1 + w_{m,2} (mod 2^f)$;
- $\Delta_m^{(2)} := m_2 - m_0 + \rho_2 + w_{m,2} (mod 2^f)$;
- $\Delta_m^{(3)} := m_3 - m_0 - w_{m,2} (mod 2^f)$;
- $\Delta_m^{(4)} := m_4 - m_0 - w_{m,2} (mod 2^f)$.

It is straightforward that the distribution of $\{\Delta_m^{(j)}\}_{j \in [4]}$ are uniformly random under the condition $\Delta_m := \sum_{j=1}^4 \Delta_m^{(j)} = m_1 - m_0 + \rho_2$, where $\rho_2$ is used to generate the DCF keys $\mathcal{K}_2 := \text{DCF}^{\text{Gen}^\text{IC}(1^\lambda, 2^f, 1, Z_{2^f}, Z_{2^f}, \rho_2)}$. Whereas $\Delta_m := \rho_2$ in the hybrid $\mathcal{H}_2$, we can show that if there exists an adversary $A$ who can distinguish the view of $\mathcal{H}_2$ from the view of $\mathcal{H}_1$ then we can construct an adversary $B$ who uses $A$ in a blackbox fashion can break $\text{DCF}^{\text{Gen}^\text{IC}(1^\lambda, 2^f, 1, Z_{2^f}, Z_{2^f}, \rho_2)}$ with the same advantage. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage

$\varepsilon_2 := \text{Adv}_{\text{DCF}^\text{IC}}^{Z_{2^f}, Z_{2^f}, t_1}(1^\lambda, A)$.

The adversary’s view of $\mathcal{H}_2$ is identical to the simulated view $\text{Exec}_{f_{\alpha, \beta}, t_1}^\text{IC}(D, S)$, Therefore, the overall distinguishing advantage is

$8 \cdot \text{Adv}_{\text{PRF}^2(Z_{2^f}, \lambda, \mathcal{A})} + 3 \cdot \text{Adv}_{\text{PRF}^2(Z_{2^f}, \lambda, \mathcal{A})} + \text{Adv}_{\text{DCF}^\text{IC}}^{Z_{2^f}, Z_{2^f}, t_1}(1^\lambda, A)$.

This concludes the proof.

C Proof of Theorem 3

Theorem 3. Let $\text{DFP}_{\alpha, \beta}^{2^f, 2^f}$ be a secure function secret sharing scheme for offset comparison function $f_{\alpha, \beta}(x) : Z_{2^f} \rightarrow Z_{2^f}$, with adversarial advantage $\text{Adv}_{\text{DCF}^\text{IC}}^{2^f, 2^f}(1^\lambda, A)$. Let $\text{PRF}_{\alpha, \beta}^{2^f} : \{0, 1\}^\lambda \times \{0, 1\}^n \rightarrow Z_{2^f}$ be a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}^\text{IC}}^{2^f}(1^\lambda, A)$. Let $\text{PRF}_{\alpha, \beta}^{2^f} : \{0, 1\}^\lambda \times \{0, 1\}^n \rightarrow Z_{2^f}$ be a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}^\text{IC}}^{2^f}(1^\lambda, A)$.

Proof. To prove Thm. 3, we construct a PPT simulator $\text{Sim}$ such that no non-uniform PPT environment $\mathcal{Z}$ can distinguish between (i) the real execution $\text{Exec}_{f_{\alpha, \beta}}(\Pi_{\text{const}}^{\text{IC}})$ where the parties $M, D, S := \{S_1, \ldots, S_4\}$ run protocol $\Pi_{\text{const}}^{\text{IC}}$, and (ii) the ideal execution $\text{Exec}_{f_{\alpha, \beta}, \text{Sim}}(D, S)$ where the parties $M, D, S_1, \ldots, S_4$ interact with functionality $f_{\alpha, \beta}^\lambda$ in the ideal world, and corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$, and (ii) the ideal execution $\text{Exec}_{f_{\alpha, \beta}, \text{Sim}}(D, S)$ where the parties $M, D, S_1, \ldots, S_4$ interact with functionality $f_{\alpha, \beta}^\lambda$ in the ideal world, and corrupted parties are controlled by the simulator $\text{Sim}$. We consider following cases.

Case 1: $\mathcal{S}_1$ (or $\mathcal{S}_2$) is corrupted.

Simulator. The simulator $\text{Sim}$ internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Sim simulates the interface of $f_{\alpha, \beta}$ as well as honest parties $M, D, S_2, S_3, S_4$. In addition, the simulator $\text{Sim}$ simulates the following interactions with $\mathcal{A}$.

- Upon receiving $\text{MODEL}, \text{sid}, M, (m, d)$ from the external $f_{\alpha, \beta}^\lambda$, the simulator $\text{Sim}$ computes $m_* := 2d - 1$ and acts as the honest model owner $M$ to do:
- for $i := 0$ to $m_* - 1$:
  * Set $k_i^{(1)}, k_i^{(2)} \leftarrow \mathbb{Z}_n$;
  * Set $t_i^{(1)}, t_i^{(2)} \leftarrow \mathbb{Z}_{2^f}$;
  * Set $p_i^{(1)} := \{k_i^{(1)}, t_i^{(1)}\}$, $p_i^{(2)} := \{k_i^{(2)}, t_i^{(2)}\}$;
- for $i := 0$ to $m_* - 1$:
  * $v_i^{(1)}, v_i^{(2)} \leftarrow \mathbb{Z}_{2^f}$;
  * Send $\{p_i^{(1)}, v_i^{(1)}\}$ to $S_1$, $\{p_i^{(2)}, v_i^{(2)}\}$ to $S_2$.
- Upon receiving $\text{DATA}, \text{sid}, D, n$ from the external $f_{\alpha, \beta}^\lambda$, the simulator $\text{Sim}$ acts as the honest data owner $D$ to do:
- for $i := 0$ to $n - 1$ do:
  * Generate $x_i^{(1)}$, $x_i^{(3)} \leftarrow \mathbb{Z}_{2^f}$, set $x_i^{(2)} := x_i^{(1)}$, $x_i^{(4)} := x_i^{(2)}$;
  * Send $x_i^{(j)}$ to $S_j$, $j \in \{4\}$.
- Upon initialization, the simulator $\text{Sim}$ acts as the honest party $\mathcal{S}_1$ to do:
- for $i := 0$ to $m_* - 1$:
  * Generate $\rho_i^{(1)}, \rho_i^{(2)} \leftarrow \mathbb{Z}_{2^f}$ and set $\rho_i^{(1)} := \rho_i^{(2)}$;
• $\mathcal{K}^{(1)}_{\lambda, \psi}, \mathcal{X}^{(2)}_{\lambda, \psi} \leftarrow \text{DCF.Gen}^\mathcal{IC}((1^\lambda, 2^{\ell - 1}, 1, \mathbb{Z}_{2^t}, \mathbb{Z}_{2^k}, \rho_1)$;

  – Send $(\mathcal{K}^{(1)}_{\lambda, \psi})_{i \in \mathbb{Z}_m}$ to $S_1$, $(\mathcal{X}^{(2)}_{\lambda, \psi})_{i \in \mathbb{Z}_m}$ to $S_2$;

• The simulator Sim does:
  – For $i \in \mathbb{Z}_m$, pick random $w_i^{(1)}, w_i^{(2)} \leftarrow \mathbb{Z}_{2^k}$ and $r_i \leftarrow \mathbb{Z}_{2^k}$;
  – Pick random $\delta \leftarrow \mathbb{Z}_{m+1}$;
  – For $i \in \mathbb{Z}_m$, set $w_i^{(3)} := w_i^{(1)}$ and $w_i^{(4)} := w_i^{(2)}$;

• Upon receiving $(\text{Eval}, \text{sid}, S_j)$ for an honest party $S_j$, $j \in \{1, 2\}$ from the external $\mathcal{F}_\mathcal{BP}$, Sim does:
  – for $i := 0$ to $m - 1$:
    * Set $\tilde{k}_i^{(j)} := 0$;
    * Send $(\text{FETCH}, \text{sid}, x_i^{(j)}, \tilde{k}_i^{(j)})$ to $\mathcal{F}_\mathcal{Sat}$ to get $x_i^{(j)}$;
    * Set $\Delta_i^{(j)} := w_i^{(j)} - x_i^{(j)} \mod (2^k)$;
    – Send $\Delta_i^{(j)}$ to $S_{3-j}$ on behalf of the honest party $S_j$;
  – Upon receiving $\Delta^{(3, j)}$ from $S_{3-j}, \Delta^{(3)}$ from $S_3$, and $\Delta^{(4)}$ from $S_4$, for an honest party $S_j, j \in \{1, 2\}$, Sim does:
    – for $i := 0$ to $m - 1$:
      * Set $\Delta_i := \sum_{j=1}^4 \Delta_i^{(q)} \mod (2^k)$;
      * Set $b_i^{(j)} := \text{DCF.Eval}^\mathcal{IC}(j, \mathcal{K}^{(j)}_{\lambda, \psi}, \Delta_i)$;
      * Set $e_i^{(j)} := (j - 1) - b_i^{(j)} \mod (2^k)$;
      * Set $e_i^{(j+1)} := b_i^{(j)} \mod (2^k)$;
    – for $i := 0$ to $m - 1$:
      * Sum up the share of edge costs along $i$-th leaf node’s path to get $c_i^{(j)}$, set $c_i^{(j)} := c_i^{(j)} - \delta \mod (m+1)$;
      * Set $\tilde{v}_i^{(j)} \leftarrow \mathbb{Z}_{2^k}, \tilde{v}_i^{(j)} := \tilde{v}_i^{(j)} \mod (m+1)$, $\tilde{v}_i^{(j)} := \tilde{v}_i^{(j)} \mod (2^k)$;
      * Send $(c_i^{(j)}, \tilde{v}_i^{(j)})_{i \in \mathbb{Z}_m}$ to $S_3$, $(c_i^{(j)}, \tilde{v}_i^{(j)})_{i \in \mathbb{Z}_m}$ to $S_4$;

**Indistinguishability.** We assume that the parties $S_1, \ldots, S_4$ communicate with each other via the secure channel functionality $\mathcal{F}_\mathcal{Sat}$ (omitted in the protocol description for simplicity). The indistinguishability is proven through a series of hybrid worlds $H_0, \ldots, H_{10}$.

**Hybrid $H_0$:** It is the real execution $\text{Exec}_{\text{Set}}^{\text{Set}}(\text{H}_\mathcal{BP}, \text{Eval}_{\mathcal{IC}}, \mathcal{A}, \mathcal{Z}, \mathcal{Z}'_\mathcal{BP})$.

**Hybrid $H_1$:** $H_1$ is the same as $H_0$ except that in $H_1$, \{ $w_i^{(1)} \}_{i \in \mathbb{Z}_m}, \{ w_i^{(2)} \}_{i \in \mathbb{Z}_m} \text{ and } \{ \mathcal{K}^{(1)}_{\lambda, \psi} \}_{i \in \mathbb{Z}_m}$ are picked uniformly random from $\mathbb{Z}_{2^k}$ instead of calculating from $\text{PRF}^Z_{\mathcal{BP}}$; \{ $r_i \}_{i \in \mathbb{Z}_m}$ is picked uniformly random from $\mathbb{Z}_{2^k}$ instead of calculating from $\text{PRF}^Z_{\mathcal{BP}}$. Set $w_i^{(3)} := w_i^{(1)}, w_i^{(4)} := w_i^{(2)}$, $i \in \mathbb{Z}_m$.

**Claim 10.** If $\text{PRF}^Z_{\mathcal{BP}} : \{0, 1\}^\lambda \times \{0, 1\}^m \rightarrow \mathbb{Z}_{2^k}$ is a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}}^Z(\mathcal{A})$, $\text{PRF}^Z_{\mathcal{BP}} : \{0, 1\} \times \{0, 1\}^m \rightarrow \mathbb{Z}_{2^k}$ is a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}}^Z(\mathcal{A})$, then $H_0$ and $H_1$ are indistinguishable with advantage $\varepsilon_1 := 3m_e \cdot \text{Adv}_{\text{PRF}}^Z(\mathcal{A}) + (m_e + 1) \cdot \text{Adv}_{\text{PRF}}^Z(\mathcal{A})$.

**Proof.** We have changed $3m_e \cdot \text{PRF}^Z_{\mathcal{BP}}$ outputs and $(m_e + 1) \cdot \text{PRF}^Z_{\mathcal{BP}}$ outputs to uniformly random strings; therefore, the overall advantage is $2m_e \cdot \text{Adv}_{\text{PRF}}^Z(\mathcal{A}) + (m_e + 1) \cdot \text{Adv}_{\text{PRF}}^Z(\mathcal{A})$ by hybrid argument via reduction.

**Hybrid $H_2$:** $H_2$ is the same as $H_0$ except that in $H_2$:

- For $j \in \{1, 2\}$:
  - Set $\Delta_i^{(j)} := w_i^{(j)} - x_i^{(j)} \mod (2^k), i \in \mathbb{Z}_m$;

**Claim 11.** If $\text{DCF.Gen}^Z_{\mathcal{BP}} : (\text{Gen}^\mathcal{IC}, \text{Eval}^\mathcal{IC}) \rightarrow \mathbb{Z}_{2^k}$ is a secure function secret sharing scheme for offset comparison function $f^\mathcal{IC}_{\mathcal{BP}}(x) : \mathbb{Z}_{2^k} \rightarrow \mathbb{Z}_{2^k}$ with adversarial advantage $\text{Adv}_{\text{DCF.Gen}^Z_{\mathcal{BP}}}(\mathcal{A})$, then $H_0$ and $H_2$ are indistinguishable with advantage $\varepsilon_2 := m_e \cdot \text{Adv}_{\text{DCF.Gen}^Z_{\mathcal{BP}}}(\mathcal{A})$.

**Proof.** Note that the corrupted party $S_1$ only sees $\{\Delta_i^{(j)}\}_{j \in \{4\}}$; therefore, the modification of $\{\Delta_i^{(j)}\}_{j \in \{4\}}$ is oblivious to $S_1$. In the hybrid $H_2$, we have $i \in \mathbb{Z}_m$:

- $\Delta_i^{(1)} := \Delta_i^{(1)} := t_i^{(1)} - x_i^{(1)} + w_i^{(1)} \mod (2^k)$;
- $\Delta_i^{(2)} := \Delta_i^{(2)} := t_i^{(2)} - x_i^{(2)} + w_i^{(2)} \mod (2^k)$;
- $\Delta_i^{(3)} := \Delta_i^{(3)} := r_i^{(3)} - x_i^{(3)} \mod (2^k)$;
- $\Delta_i^{(4)} := \Delta_i^{(4)} := r_i^{(4)} - x_i^{(4)} \mod (2^k)$;

It is straightforward that for $i \in \mathbb{Z}_m$, the distribution of $\{\Delta_i^{(j)}\}_{j \in \{4\}}$ are uniformly random under the condition $\Delta_i := \sum_{j=1}^4 \Delta_i^{(j)} = t_i - x_i + \rho_1$, where $\rho_1$ is used to generate the DCF keys $\mathcal{K}^{(1)}_{\lambda, \psi}, \mathcal{K}^{(2)}_{\lambda, \psi} \leftarrow \text{DCF.Gen}^\mathcal{IC}(1^\lambda, 2^{\ell - 1}, 1, \mathbb{Z}_{2^k}, \mathbb{Z}_{2^k}, \rho_1)$.

Whereas $\Delta_i := \rho_i - x_i$ in the hybrid $H_2$, we can show that if there exists an adversary $\mathcal{A}$ who can distinguish the view of $H_2$ from the view of $H_0$ then we can construct an adversary $\mathcal{B}$ who uses $\mathcal{A}$ in a blackbox fashion can break $\text{DCF.Gen}^Z_{\mathcal{BP}} := (\text{Gen}^\mathcal{IC}, \text{Eval}^\mathcal{IC})$ with the same advantage. Therefore, $H_2$ and $H_0$ are indistinguishable with adversarial advantage $\varepsilon_2 := m_e \cdot \text{Adv}_{\text{DCF.Gen}^Z_{\mathcal{BP}}}(\mathcal{A})$.

**Hybrid $H_3$:** $H_3$ is the same as $H_2$ except that in $H_3$:

- For $j \in \{1, 2\}$:
  - Set $\tilde{k}_i^{(j)} := 0, i \in \mathbb{Z}_m$;

instead of

- For $j \in \{1, 2\}$:
  - Set $\tilde{k}_i^{(j)} := 0, i \in \mathbb{Z}_m$;
• Set $k_i^{(j)} := k_i^{(j)}$, $i \in \mathbb{Z}_{m_i}$.

Claim 12. $\mathcal{H}_R$ and $\mathcal{H}_I$ are perfectly indistinguishable.

Proof. Since $\{k_i^{(j)}\}_{i \in \mathbb{Z}_{m_i}, j \in \{2\}}$ are only sent to the simulated $\mathcal{F}_{\text{set}}^d$, which is oblivious to $S_1$, $\mathcal{H}_R$ and $\mathcal{H}_I$ are perfectly indistinguishable.

The adversary’s view of $\mathcal{H}_I$ is identical to the simulated view $\text{Exec}_{\mathcal{F}_{\text{bp}}^4, \mathcal{Z}}$. Therefore, the overall distinguishing advantage is

$$3m_c \cdot \text{Adv}_{\text{PRF}^Z_{\mathcal{Z}}}(1^\lambda, \mathcal{A}) + (m_c + 1) \cdot \text{Adv}_{\text{PRF}^Z_{\mathcal{Z}}}(1^\lambda, \mathcal{A})$$

$$+ m_c \cdot \text{Adv}_{\text{DCFI}_C}(1^\lambda, \mathcal{A})$$

Case 2: $S_3$ (or $S_4$) is corrupted.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Simulates the interface of $\mathcal{F}_{\text{set}}$ as well as honest parties $M, D, S_2, S_3, S_4$. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$:

• Upon receiving (MODEL, $\text{sid}, M, (m, d)$) from the external $\mathcal{F}_{\text{bp}}^4$, the simulator Sim computes $m_c := 2^d - 1$ and acts as the honest model owner $M$ to do:
  - for $i := 0$ to $m_c - 1$:
    * Set $k_i^{(1)}, k_i^{(2)} \leftarrow \mathbb{Z}_m$;
    * Set $l_i^{(1)}, l_i^{(2)} \leftarrow \mathbb{Z}_m$;
    * Set $P_i := \{k_i^{(1)}, l_i^{(1)}\}, P_i^{(2)} := \{k_i^{(2)}, l_i^{(2)}\}$;
  - for $i := 0$ to $m_c$:
    * $v_i^{(1)}, v_i^{(2)} \leftarrow \mathbb{Z}_m$;
    * Send $(P_i^{(1)}, v_i^{(1)})$ to $S_1$, $(P_i^{(2)}, v_i^{(2)})$ to $S_2$.

• Upon receiving (DATA, $\text{sid}, D, n$) from the external $\mathcal{F}_{\text{bp}}^4$, the simulator Sim acts as the honest data owner $D$ to do:
  - for $i := 0$ to $n - 1$ do:
    * Generate $x_i^{(1)}, x_i^{(3)} \leftarrow \mathbb{Z}_d$, set $x_i^{(2)} := x_i^{(1)}, x_i^{(4)} := x_i^{(3)}$;
    * Send $x_i^{(3)}$ to $S_j$, $j \in \{4\}$.

• Upon receiving (Eval, $\text{sid}, S_j$) for an honest party $S_j$, $j \in \{1, 2\}$ from the external $\mathcal{F}_{\text{bp}}^4$, Sim does:
  - for $i := 0$ to $m_c - 1$:
    * Set $k_i^{(j)} := 0$;
    * Send $(\text{FETCH}, \text{sid}, x_i^{(3)}, k_i^{(j)})$ to $\mathcal{F}_{\text{set}}^d$ to get $x_i^{(j)}$;
    * Set $\Delta x_i^{(j)} := w_i^{(j)} - x_i^{(j)}$ (mod $2^d$);
    * Send $\Delta x_i^{(j)}$ to $S_{3-j}$ on behalf of the honest party $S_j$;

• Upon receiving $(\text{Eval}, \text{sid}, S_j)$ for an honest party $S_j$, $j \in \{3, 4\}$ from the external $\mathcal{F}_{\text{bp}}^4$, Sim does:
  - for $i := 0$ to $m_c - 1$:
    * Set $k_i^{(j)} := 0$;
    * Send $(\text{FETCH}, \text{sid}, x_i^{(3)}, k_i^{(j)})$ to $\mathcal{F}_{\text{set}}^d$ to get $x_i^{(j)}$;
    * Set $\Delta x_i^{(j)} := p_i^{(j-2)} - x_i^{(j)} - w_i^{(j)}$ (mod $2^d$);
    * Send $\Delta x_i^{(j)}$ to $S_1$ and $S_2$ on behalf of the honest party $S_j$;

• Upon receiving $\Delta^{(3-j)}$ from $S_{3-j}$, $((\mathcal{F}_{\text{set}}^d)^{\bullet})_{\in \mathbb{Z}_m}, \Delta x^{(3)}$ from $S_1$, and $\Delta x^{(4)}$ from $S_4$, for an honest party $S_j$, $j \in \{1, 2\}$, Sim does:
  - for $i := 0$ to $m_c - 1$:
    * Set $\Delta x_i := \sum_{j=1}^{m_c} \Delta x_i^{(j)}$ (mod $2^d$);
    * Set $b_i^{(j)} \leftarrow \text{DCF.Eval}^C(j, x_i^{(j)}, \Delta x_i)$;
    * Set $e_i^{(j)} := (j - 1 - b_i^{(j)}) \cdot r_i$ (mod $2^d$);
    * Set $e_i^{(j)} := b_i^{(j)} \cdot r_i$ (mod $2^d$);
  - for $i := 0$ to $m_c$:
    * Sum up the share of edge costs along $i$-th leaf node’s path to get $c_i^{(j)}$, set $\hat{c}_i^{(j)} := c_i^{(j)}$ (mod $m_c+1$);
    * Set $\hat{v}_i := \mathbb{Z}_d^*$, $\hat{v}_i^{(j)} := c_i^{(j)}$ (mod $m_c+1$) $- \hat{v}_i$ (mod $2^d$);
    * Send $(\hat{c}_i^{(j)}, \hat{v}_i^{(j)}) \in \mathbb{Z}_{m_c+1}$ to $S_3$, $(\hat{c}_i^{(j)}, \hat{v}_i^{(j)}) \in \mathbb{Z}_{m_c+1}$ to $S_4$.

Indistinguishability. We assume that the parties $S_1, \ldots, S_4$ communicate with each other via the secure channel functionality $\mathcal{F}_{\text{sec}}$ (omitted in the protocol description for simplicity). The indistinguishability is proven through a series of hybrid worlds $\mathcal{H}_0, \ldots, \mathcal{H}_i$.

Hybrid $\mathcal{H}_0$: It is the real protocol execution $\text{Exec}_{\mathcal{F}_{\text{set}}^d, \mathcal{F}_{\text{bp}}^4, \mathcal{Z}}$.

Hybrid $\mathcal{H}_i$: $\mathcal{H}_i$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_i$, $\{w_i^{(1)}\}_{i \in \mathbb{Z}_m}, \{w_i^{(2)}\}_{i \in \mathbb{Z}_m}$ and $\{P_i^{(2)}\}_{i \in \mathbb{Z}_m}$ are picked uniformly random from $\mathbb{Z}_d^*$ instead of calculating from $\text{PRF}^Z_{\mathcal{Z}}$; $\{r_i\}_{i \in \mathbb{Z}_{m_c+1}}$ is picked uniformly random from $\mathbb{Z}_d^*$ instead of calculating from $\text{PRF}^Z_{\mathcal{Z}}$. Set $w_i^{(3)} := w_i^{(1)}, w_i^{(4)} := w_i^{(2)}$, $i \in \mathbb{Z}_{m_c}$.

Claim 13. If $\text{PRF}^Z_{\mathcal{Z}} : \{0, 1\}^\lambda \times \{0, 1\}^m \rightarrow \mathbb{Z}_d^*$ is a secure pseudorandom function with adversarial advantage $\text{Adv}_{\text{PRF}^Z_{\mathcal{Z}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_0$ and $\mathcal{H}_i$ are indistinguishable with advantage $\epsilon_1 := 3m_c \cdot \text{Adv}_{\text{PRF}^Z_{\mathcal{Z}}}(1^\lambda, \mathcal{A}) + (m_c + 1) \cdot \text{Adv}_{\text{PRF}^Z_{\mathcal{Z}}}(1^\lambda, \mathcal{A})$.

Proof. We have changed $2m_c$ $\text{PRF}^Z_{\mathcal{Z}}$ outputs and $(m_c + 1)$ $\text{PRF}^Z_{\mathcal{Z}}$ outputs to uniformly random strings; therefore, the overall advantage is $3m_c \cdot \text{Adv}_{\text{PRF}^Z_{\mathcal{Z}}}(1^\lambda, \mathcal{A}) + (m_c + 1) \cdot \text{Adv}_{\text{PRF}^Z_{\mathcal{Z}}}(1^\lambda, \mathcal{A})$ by hybrid argument via reduction.

Hybrid $\mathcal{H}_2$: $\mathcal{H}_2$ is the same as $\mathcal{H}_i$ except that in $\mathcal{H}_i$:

For $j \in \{1, 2\}$:

• Set $k_i^{(j)} := 0, i \in \mathbb{Z}_{m_i}$

Instead of

For $j \in \{1, 2\}$:

• Set $k_i^{(j)} := k_i^{(j)}$, $i \in \mathbb{Z}_{m_i}$.

Claim 14. $\mathcal{H}_0$ and $\mathcal{H}_i$ are perfectly indistinguishable.
Proof. Since $(k^{(j)}_{i})_{j \in \{1,2,3\}}$ are only sent to the simulated $\mathcal{G}_{\text{bp}}^{\mathcal{A},\mathcal{Z}}$, which is oblivious to $S_1$, $\mathcal{H}_2$ and $\mathcal{H}_1$ are perfectly indistinguishable.

The adversary’s view of $\mathcal{H}_2$ is identical to the simulated view $\mathcal{G}_{\text{bp}}^{\mathcal{A},\mathcal{Z}}$. Therefore, the overall distinguishing advantage is

$$3m_{\text{c}} \cdot \text{Adv}_{\text{PRF}}^{\mathcal{Z}}(1^{\lambda}, \mathcal{A}) + (m_{\text{c}} + 1) \cdot \text{Adv}_{\text{PRF}}^{\mathcal{Z}, \mathcal{A}}(1^{\lambda}, \mathcal{A})$$

This concludes the proof.

D Proof of Theorem 4

Theorem 4. The protocol $\Pi_{\text{eval}}^{\text{poly}}$ as described in Fig. 15 and $\Pi_{\text{eval}}^{\text{poly}}$ as described in Fig. 16 UC-realizes $\mathcal{G}_{\text{bp}}^{\mathcal{A},\mathcal{Z}}$ as described in Fig. 2 in the $(\mathcal{F}_{\text{bp}}, \mathcal{F}_{\text{eval}})$-hybrid model against semi-honest adversaries who can statically corrupted up to 1 server.

Proof. To prove Thm. 4, we construct a PPT simulator $\text{Sim}$ such that no non-uniform PPT environment $\mathcal{Z}$ can distinguish between (i) the real execution $\mathcal{G}_{\text{bp}}^{\mathcal{A},\mathcal{Z}}$ of the protocol $\Pi_{\text{eval}}^{\text{poly}}$, where the parties $M, D, S := \{S_1, \ldots, S_4\}$ run protocol $\Pi_{\text{eval}}^{\text{poly}}$, in the $(\mathcal{F}_{\text{bp}}, \mathcal{F}_{\text{eval}})$-hybrid world and the corrupted parties are controlled by a dummy adversary $A$ who simply forwards messages from/to $\mathcal{Z}$, and (ii) the ideal execution $\mathcal{G}_{\text{bp}}^{\mathcal{A},\text{Sim},\mathcal{Z}}$ where the parties $M, D, S_1, \ldots, S_4$ interact with functionality $\mathcal{F}_{\text{bp}}^{\mathcal{A}}$ in the ideal world, and corrupted parties are controlled by the simulator $\text{Sim}$.

**Simulator.** The simulator $\text{Sim}$ internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. $\text{Sim}$ simulates the interface of $(\mathcal{F}_{\text{bp}}, \mathcal{F}_{\text{eval}})$ as well as honest parties $M, D, S_2, S_3, S_4$. In addition, the simulator $\text{Sim}$ simulates the following interactions with $\mathcal{A}$.

- Upon receiving $(\text{MODEL}, \text{sid}, M, (m, d))$ from the external $\mathcal{F}_{\text{bp}}^{\mathcal{A}}$, the simulator $\text{Sim}$ acts as the honest model encoder $M$ to do:
  - Build the position mapping, using dummy elements as $P_i := \{j_{\text{left}} := 0, j_{\text{right}} := 0, g_{\text{left}} := 0, g_{\text{right}} := 0, t_i := 0, v_i := 0\}$.
  - For $i := 0$ to $m - 1$ do:
    - Set $I_{\text{left}}(1) := Z_m$, $S_{\text{left}}(2) := I_{\text{left}}(1)$.
    - Set $I_{\text{left}}(3) := I_{\text{left}}(4) := I_{\text{left}}(1)$ (mod $m$).
    - Set $I_{\text{right}}(1) := Z_m$, $S_{\text{right}}(2) := I_{\text{right}}(1)$.
    - Set $I_{\text{right}}(3) := I_{\text{right}}(4) := I_{\text{right}}(1)$ (mod $m$).
    - Set $I_{\text{left}}(1) := Z_m$, $S_{\text{left}}(2) := I_{\text{left}}(1)$.
    - Set $I_{\text{right}}(1) := Z_m$, $S_{\text{right}}(2) := I_{\text{right}}(1)$.
    - Set $S_{\text{left}}(1) := t_i$, $S_{\text{right}}(1) := t_i$.
    - Set $S_{\text{left}}(2) := t_i$, $S_{\text{right}}(2) := t_i$.
    - Set $S_{\text{left}}(3) := t_i$, $S_{\text{right}}(3) := t_i$.
    - Set $S_{\text{left}}(4) := t_i$, $S_{\text{right}}(4) := t_i$.
    - Set $S_{\text{left}}(5) := t_i$, $S_{\text{right}}(5) := t_i$.
    - Set $S_{\text{left}}(6) := t_i$, $S_{\text{right}}(6) := t_i$.
  - Return $S_i$.
  - Upon receiving $(\text{DATA}, \text{sid}, D, n)$ from the external $\mathcal{F}_{\text{bp}}^{\mathcal{A}}$, the simulator $\text{Sim}$ acts as the honest data owner $D$ to do:
    - Generate $i_{\text{id}}$, $s_{\text{id}} := Z_2$, $id_i := \Sigma_{i=1}^{m_2} i_{\text{id}}(i)$ (mod $n$).
    - Send $(p^{(j)}_{\text{id}}(i), k^{(j)}_{\text{id}})$ to $S_j$, $j \in \{4\}$.
    - Upon receiving $(\text{Eval}, \text{sid}, S_j)$ for an honest party $S_j$, from the external $\mathcal{F}_{\text{bp}}^{\mathcal{A}}$ $\text{Sim}$ does:
      - For $i := 1$ to $d$ do:
        - Send $(\text{FETCH}, \text{sid}, S_j)$ to $G_{\text{bp}}^{\mathcal{A},\mathcal{Z}}$ to get $x_{i_i}$.
        - Send $(\text{FETCH}, \text{sid}, \text{PRF}(i_{\text{id}}), s_{\text{id}})$ to $G_{\text{bp}}^{\mathcal{A},\mathcal{Z}}$ to get $p_{\text{id}}^{(j)} := (I_{\text{left}}(i_{\text{id}}), I_{\text{right}}(i_{\text{id}}), J_{\text{left}}(i_{\text{id}}), J_{\text{right}}(i_{\text{id}}))$.
        - Set $res_{i} := res_{i-1} + x_{i_i}$ (mod $2^2$).
        - If $i \geq d$, return $res$ to the receiver $R$ and break.
      - Upon $\text{FETCH}$ to $\text{Sim}$, $\text{Sim}$ does:
        - Send $(\text{COMP}, \text{sid}, \text{left}(i_{\text{id}}), \text{right}(i_{\text{id}}), (x_{i_i}, t_{i_{\text{id}}}))$ to $\mathcal{F}_{\text{eval}}^{\mathcal{A},\mathcal{Z}}$.
        - Upon receiving $\mathcal{F}_{\text{eval}}^{\mathcal{A},\mathcal{Z}}$ from the corrupted party $S_j$, the simulator $\text{Sim}$ sends $(\text{Eval}, \text{sid}, s_{\text{id}})$ to the external $\mathcal{F}_{\text{bp}}^{\mathcal{A}}$.
        - When the simulated $\mathcal{F}_{\text{eval}}^{\mathcal{A},\mathcal{Z}}$ receives input from the corrupted party $S_j$, the simulator $\text{Sim}$ sends $(\text{Eval}, \text{sid}, s_{\text{id}})$ to the external $\mathcal{F}_{\text{bp}}^{\mathcal{A}}$.

Indistinguishability. We assume that the parties $M, D, S_1, \ldots, S_4$ communicate with each other via the secure channel functionality $\mathcal{F}_{\text{eval}}^{\mathcal{A}}$ (omitted in the protocol description for simplicity). The views of $\mathcal{A}$ and $\mathcal{Z}$ in $\mathcal{G}_{\text{bp}}^{\mathcal{A},\mathcal{Z}}$ and $\mathcal{G}_{\text{bp}}^{\mathcal{A},\text{Sim},\mathcal{Z}}$ are identical. Therefore, it is perfectly indistinguishable. This concludes the proof.

E Oblivious Selection $\Pi_{\text{sel}}$

In this section, we provide the details of our oblivious selection protocol invoked in our polynomial-round private decision tree protocol $\Pi_{\text{eval}}^{\text{poly}}$ in Sec. 6.2. The protocol $\Pi_{\text{sel}}$ is described in Fig. 18.
Offline phase:
- $S_1$ dose for $i = 1$ to $d$:  
  - Generate $\psi_{j,i} \leftarrow \text{Gen}(\psi_{j,i}, 1, \mathbb{Z}_q)$; 
  - $\psi_{j,i} \leftarrow \text{Gen}(\psi_{j,i}, 1, \mathbb{Z}_q)$; 
- $S_3$ dose for $i = 1$ to $d$:  
  - Generate $\psi_{j,i} \leftarrow \text{Gen}(\psi_{j,i}, 1, \mathbb{Z}_q)$; 
  - $\psi_{j,i} \leftarrow \text{Gen}(\psi_{j,i}, 1, \mathbb{Z}_q)$; 
- $S_1$ sends $(\text{Key}_{\psi_{j,i}}, N_{\psi_{j,i}}) \in [d]$ to $S_3$, $(\text{Key}_{\psi_{j,i}}, N_{\psi_{j,i}}) \in [d]$ to $S_3$, and $S_1$ sends $(\text{Key}_{\psi_{j,i}}, N_{\psi_{j,i}}) \in [d]$ to $S_7$. 
- $S_2$ sets $\psi_{j,i} = \text{PRF}_{\psi_{j,i}}(\text{sid}, 0)$, $S_3$ sets $\psi_{j,i} = 0$, and $\psi_{j,i} = 0$, for $i \in [d]$.

Online phase:
- For select $x_0$ and $p_0$:
  - Upon receiving $(\text{Fetch}, \text{sid}, x(j), \vec{K}(j))$ from the environment $Z$, player $S_j, j \in \{1, 2\}$ does:
    * Set $k_{x(j)} = \text{PRF}_{\psi_{j,i}}(\text{sid}, 1)$, $w_{x(j)} = \text{PRF}_{\psi_{j,i}}(\text{sid}, 2)$, $w_{x(j)} = \text{PRF}_{\psi_{j,i}}(\text{sid}, 3)$, $w_{x(j)} = \text{PRF}_{\psi_{j,i}}(\text{sid}, 4)$; 
    * Set $\delta_0 = k_{x(j)} + w_{x(j)}$, $\delta_0 = k_{x(j)} + w_{x(j)}$ (mod $n$), $\delta_0 = k_{x(j)} + w_{x(j)}$ (mod $\bar{m}$); 
    * Send $(\delta_0, \delta_0)$ to $S_{j-1}$, $(\delta_0, \delta_0)$ to $S_j$ and $S_0$; 
- Upon receiving $(\text{Fetch}, \text{sid}, x(j), \vec{K}(j))$ from $S_{j-1}$, $(\delta_0, \delta_0)$ from $S_j$, and $(\delta_0, \delta_0)$ from $S_0$, player $S_j, j \in \{1, 2\}$ does:
  - Set $\delta_0 = \delta_0 + \delta_0 + \delta_0$, $\delta_0 = \delta_0$ (mod $n$), $\delta_0 = \delta_0$ (mod $m$), $\delta_0 = \delta_0$ (mod $\bar{m}$) for $q \in \mathbb{Z}_q$; 
  - Set $\delta_0 = \delta_0 + \delta_0 + \delta_0 + \delta_0$ (mod $n$), $\delta_0 = \delta_0$ (mod $m$), $\delta_0 = \delta_0$ (mod $\bar{m}$) for $q \in \mathbb{Z}_q$; 
  - Set $\delta_0 = \delta_0 + \delta_0 + \delta_0$, $\delta_0 = \delta_0$ (mod $n$), $\delta_0 = \delta_0$ (mod $m$), $\delta_0 = \delta_0$ (mod $\bar{m}$); 
  - Send $(\delta_0, \delta_0)$ to $S_1$ and $S_2$, $(\delta_0, \delta_0)$ to $S_0$;
- Upon receiving $(\text{Fetch}, \text{sid}, x(j), \vec{K}(j))$ from the environment $Z$, player $S_j, j \in \{3, 4\}$ does:
  - Set $k_{x(j)} = \text{PRF}_{\psi_{j,i}}(\text{sid}, 1)$, $w_{x(j)} = \text{PRF}_{\psi_{j,i}}(\text{sid}, 2)$; 
  - Set $\delta_0 = \delta_0 - \delta_0$, $\delta_0 = \delta_0$ (mod $n$), $\delta_0 = \delta_0$ (mod $m$); 
  - Set $\delta_0 = \delta_0 - \delta_0$, $\delta_0 = \delta_0$ (mod $m$); 
  - Send $(\delta_0, \delta_0) \in [d]$ to $S_1$, $(\delta_0, \delta_0) \in [d]$ to $S_2$, and $(\delta_0, \delta_0) \in [d]$ from $S_{j-1}$ to $S_j$.

Figure 18: Selection protocol.