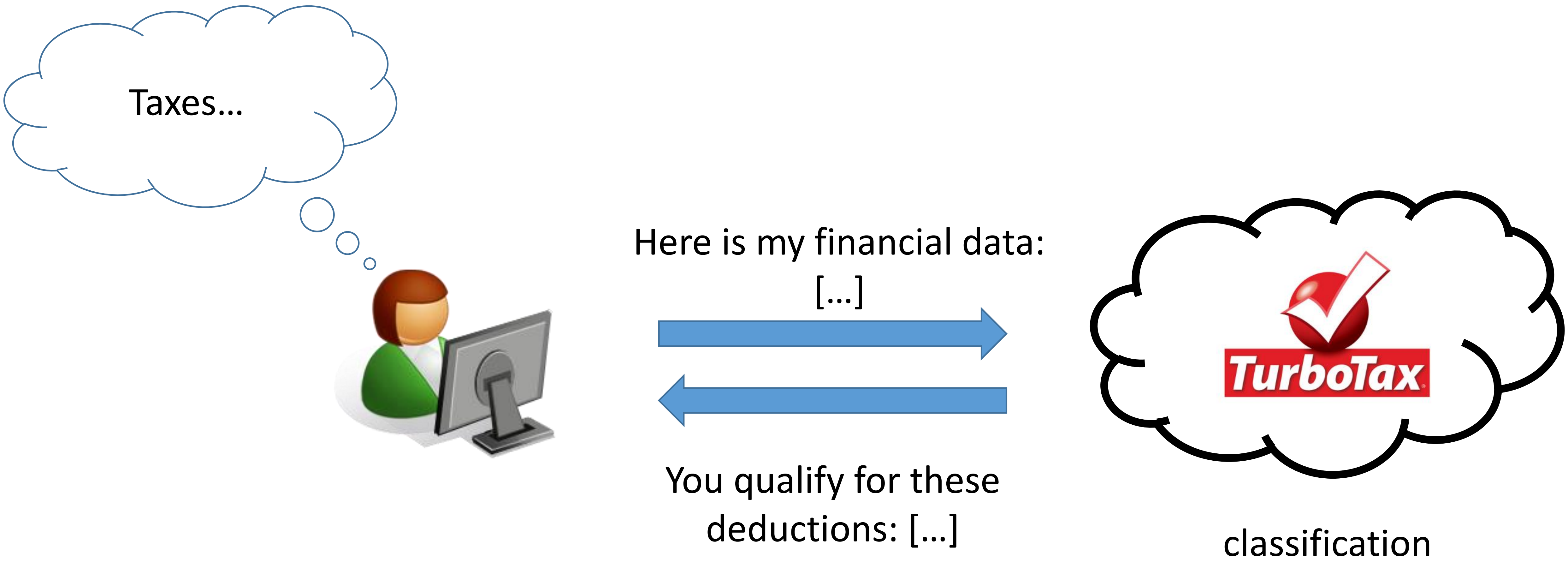
Privately Evaluating Decision Trees and Random Forests

David Wu

Joint work with Tony Feng, Michael Naehrig, and Kristin Lauter

December, 2014

Motivations



The Power of the Cloud

Advantage of the cloud: *big data*

But can now the cloud be trusted?

- Financial Records
- Medical Records
- Legal Records
- Personal Information

Privacy-Preserving Machine Learning

Leverage the power and data available in cloud-based services

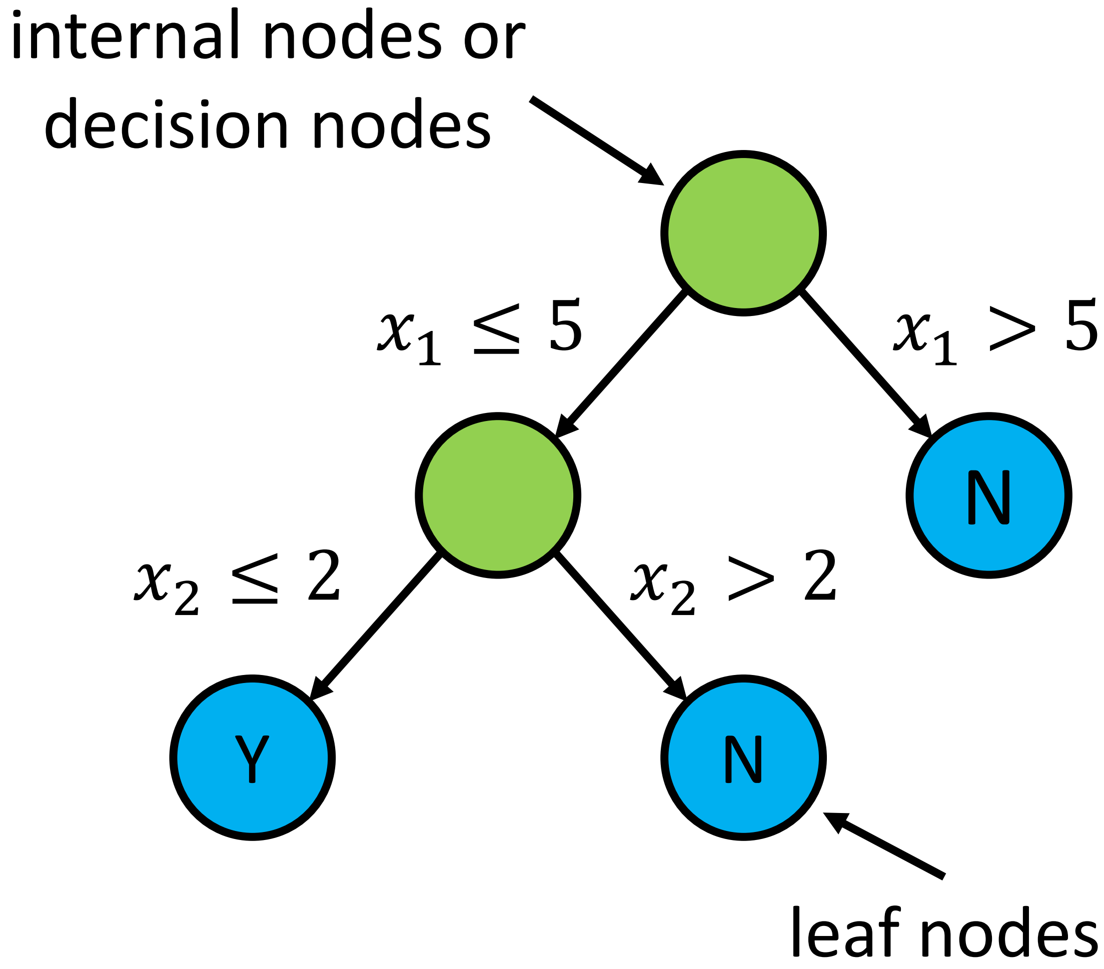
Preserve user privacy

Scope of This Talk

Consider one particular model: decision trees and their generalization, random forests

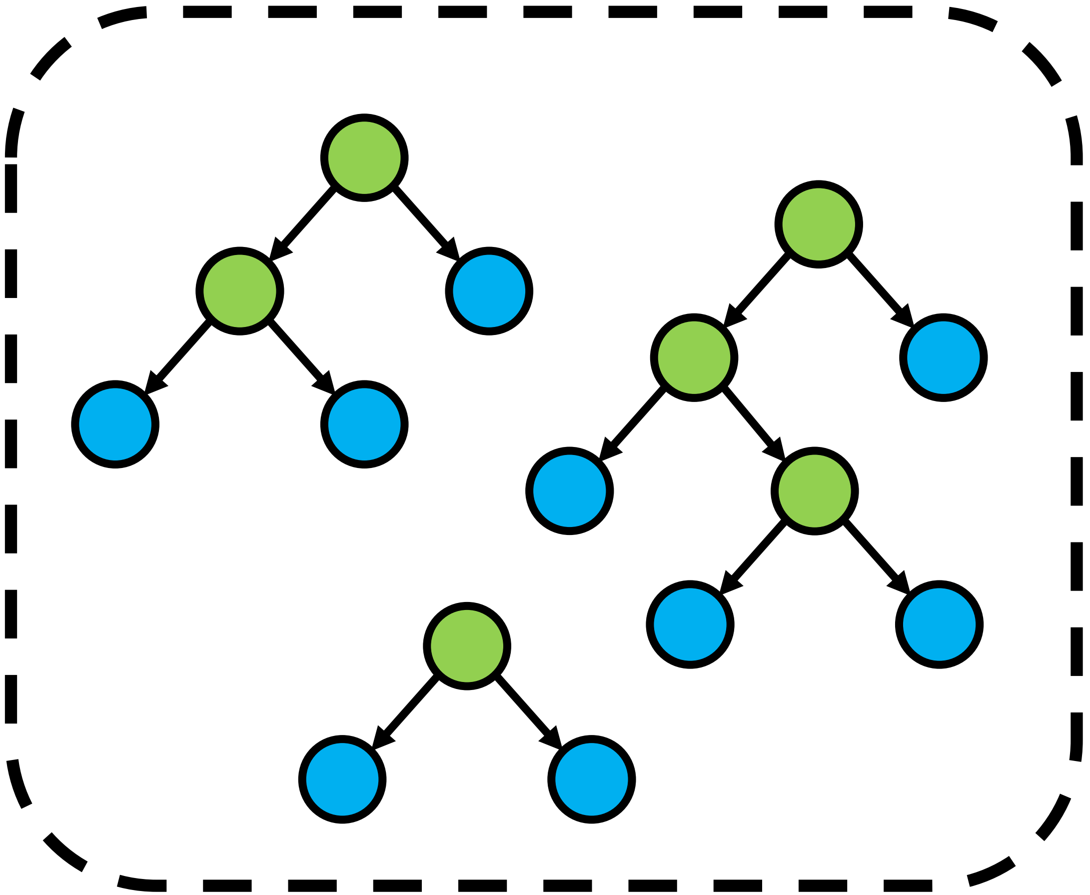
Assume that the server already has the model: focus on *private evaluation* of models

Decision Trees



- Nonlinear models for regression or classification
- Consists of a series of decision variables (tests on the feature vector)
- Evaluation corresponds to tree traversal

Random Forests



- Train many decision trees on random subsets of the features
- Output is average (majority) of outputs of individual decision trees for regression (classification)
- Reduces variance of model

Security Model

Semi-honest adversary: follow the protocol as written, but may try to learn additional information from the protocol trace (*honest-but-curious*)

Malicious adversary: can deviate arbitrarily from the protocol to satisfy its objectives

Server-Side and Client-Side Privacy

Privacy for the client: server learns no information about the client's query

Privacy for the server: client does not learn anything about the model other than what s/he already learns from the output

Formally, we use the real-world / ideal-world paradigm

Comparison Protocol

Comparison Protocol [DGK07, BPTG14]

Recall decision tree setting:

- Server has a decision tree (the model)
- Client has feature vector

Comparison Protocol [DGK07, BPTG14]

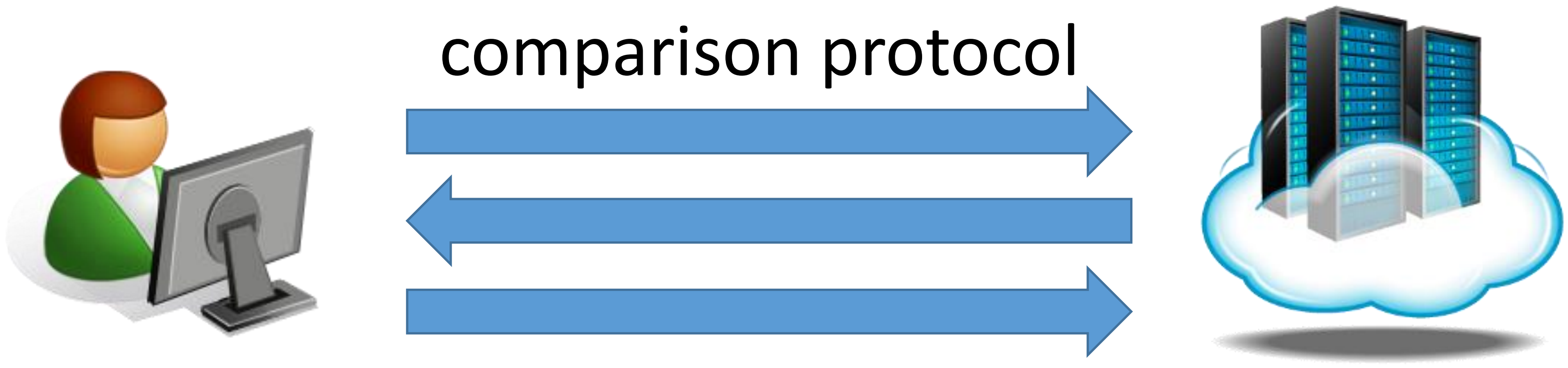
Basic building block for decision trees: evaluating comparisons of the form

$$\mathbf{1}\{x_{i_k} < t_k\}$$

index into feature vector (x_{i_k} is the client's input)

threshold (server's input)

Comparison Protocol [DGK07, BPTG14]



client input: x

server input: y

Desired functionality:

Server learns an *encryption* of comparison bit (under the client's public key), client learns nothing

Back to the Comparison Protocol…

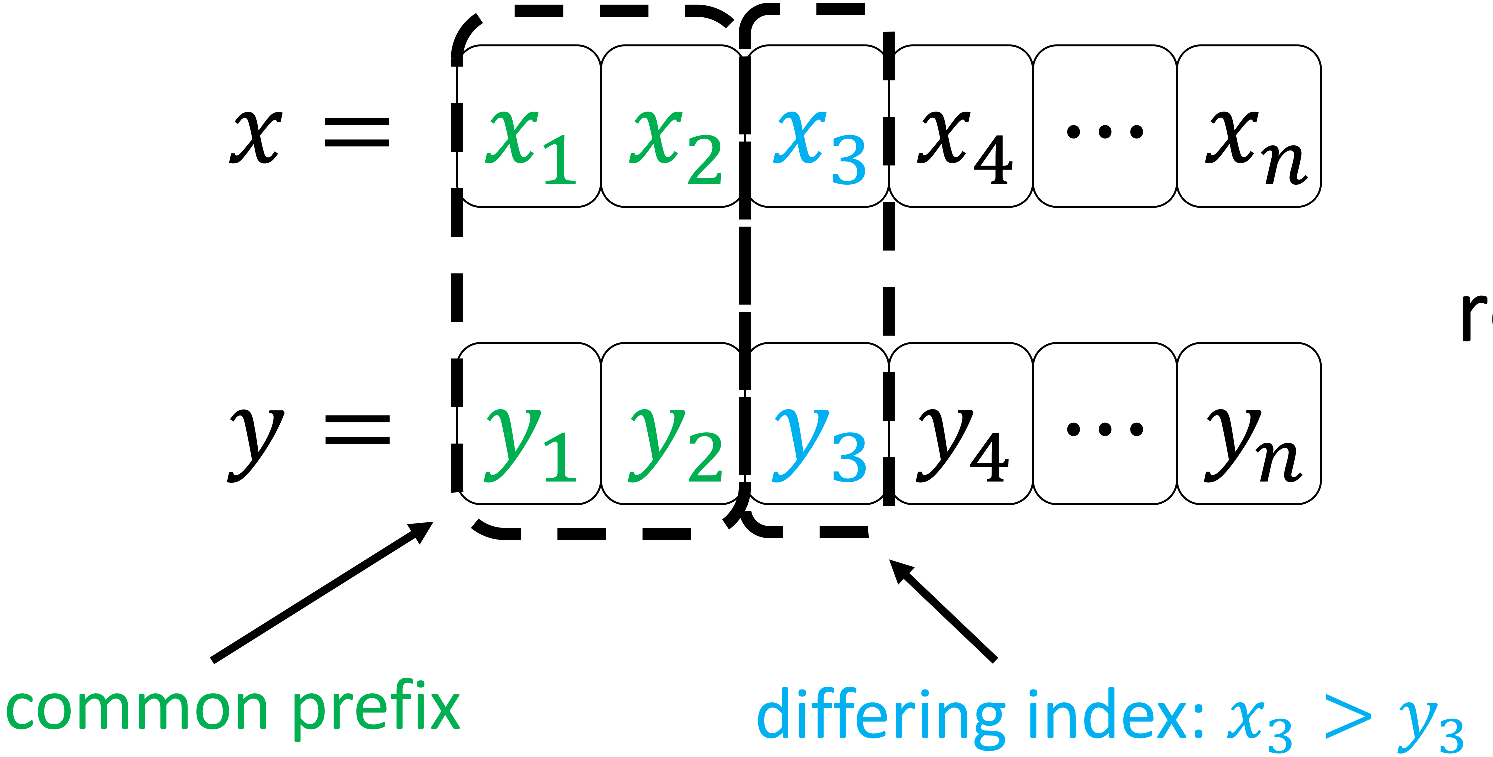
$$x = \boxed{x_1}\boxed{x_2}\boxed{x_3}\boxed{x_4}\boxed{\cdots}\boxed{x_n}$$

$$y = \boxed{y_1}\boxed{y_2}\boxed{y_3}\boxed{y_4}\boxed{\cdots}\boxed{y_n}$$

binary representations

Take two positive integers x, y and consider their binary representations

Comparison Protocol [DGK07, BPTG14]



binary representations

Observation:

$x > y$ if there is an index such that $x_i > y_i$ and for all $j < i$, $x_j = y_j$

Comparison Protocol [DGK07, BPTG14]

$\mathrm{Enc}(x_1) \cdots \mathrm{Enc}(x_n)$

client input: x

server input: y

Step 1: Client sends bitwise encryptions to server

Comparison Protocol [DGK07, BPTG14]

Step 2: Server chooses $s \xleftarrow{\$} \{-1,1\}$ and homomorphically computes

$$\mathrm{Enc}\left(x_i - y_i + s + 3\sum_{j<i}(x_j \oplus y_j)\right)$$

server input: y

Note: encryption scheme needs to be additively homomorphic

Comparison Protocol [DGK07, BPTG14]

Term server computes:

$$w_i := \boxed{x_i - y_i + s} + \boxed{3\sum_{j<i}(x_j \oplus y_j)}$$

If $s = 1$, $x_i - y_i + s = 0$ if and only if $x_i < y_i$

If $s = -1$, $x_i - y_i + s = 0$ if and only if $x_i > y_i$

Always non-negative, and if non-zero, then $w_i > 0$

Comparison Protocol [DGK07, BPTG14]

Term server computes:

$$w_i := \boxed{x_i - y_i + s} + \boxed{3 \sum_{j<i} (x_j \oplus y_j)}$$

Recall observation:

$x > y$ if and only if there is i such that $x_i > y_i$ and for all $j < i$, $x_j = y_j$

if $s = -1$, $x > y$ if and only if there exists i such that $w_i = 0$
if $s = 1$, $x < y$ if and only if there exists i such that $w_i = 0$

Comparison Protocol [DGK07, BPTG14]

$\mathrm{Enc}(w_1) \cdots \mathrm{Enc}(w_n)$

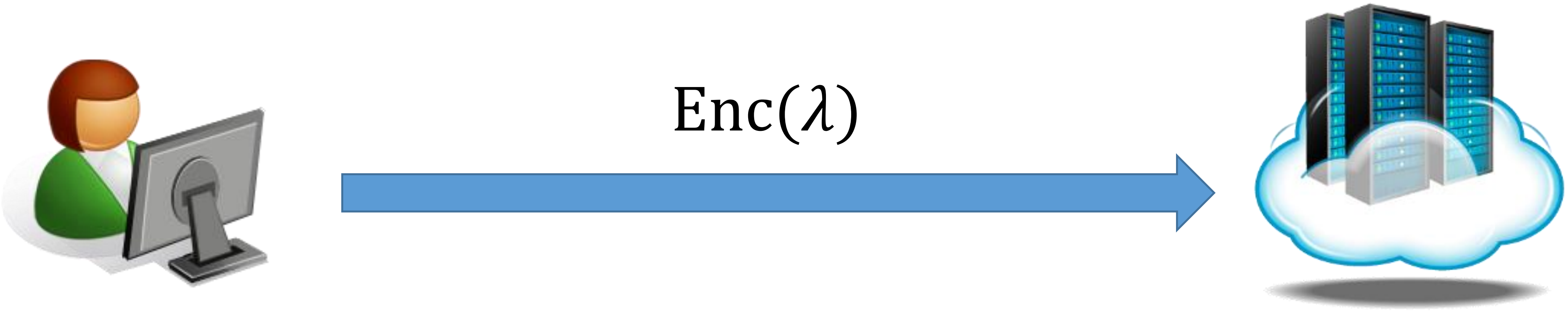
client input: x

server input: y

Step 3: Server sends back $\mathrm{Enc}(w_1) \cdots \mathrm{Enc}(w_n)$

Technical detail: Server first multiplies by a random non-zero element

Comparison Protocol [DGK07, BPTG14]



client input: x

server input: y

Step 4: Client decrypts the w_i and sends back $\mathrm{Enc}(\lambda)$ where $\lambda = 1$ only if there exists i such that $w_i = 0$ and 0 otherwise

Comparison Protocol [DGK07, BPTG14]

Step 5: Given $\mathrm{Enc}(\lambda)$ and s, server can compute result of comparison:

$$\mathrm{Enc}(\mathbf{1}\{x < y\}).$$

server input: y

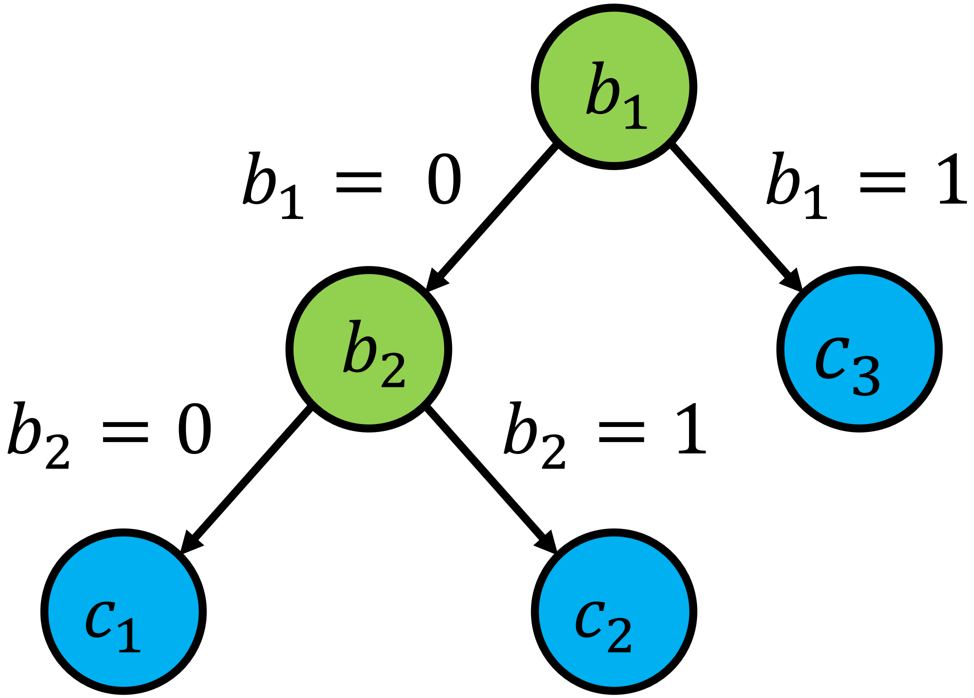
Recall:

if $s = -1$, $x > y$ if and only if there exists i such that $w_i = 0$

if $s = 1$, $x < y$ if and only if there exists i such that $w_i = 0$

Semi-honest Secure Protocol

Key Idea: suppose we give the client b_1, b_2, and the structure of the tree



Then, client can compute the *index* of the outcome

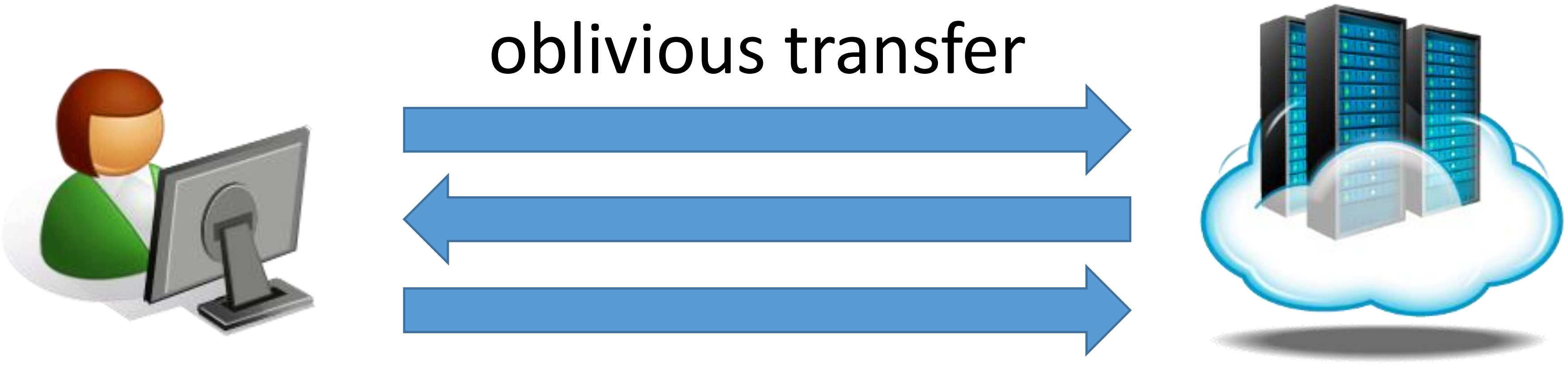
Problem: Leaks the structure of the tree!

Semi-honest Secure Protocol

Suppose client knew the index of the outcome

Problem reduces to well-studied problem: oblivious transfer

Oblivious Transfer (OT)



client's input:
index i

server's input:
database $\{m_1, \ldots, m_n\}$

Desired functionality:

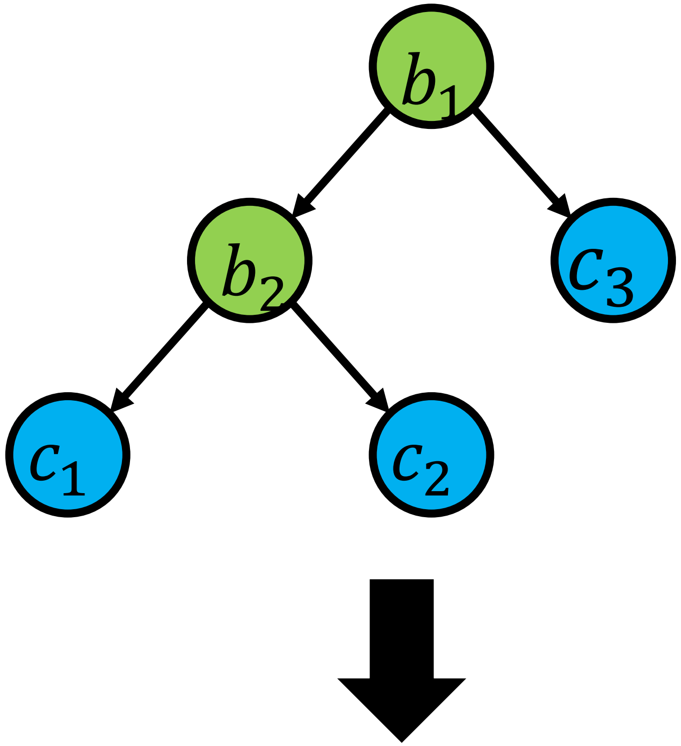
Client learns m_i and nothing else, server learns nothing

Semi-honest Secure Protocol

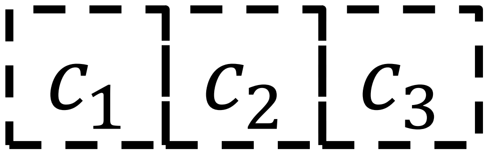
Suppose client knew the index of the outcome

Problem reduces to OT: treat leaves as database, client knows index

Problem: Need to hide structure!

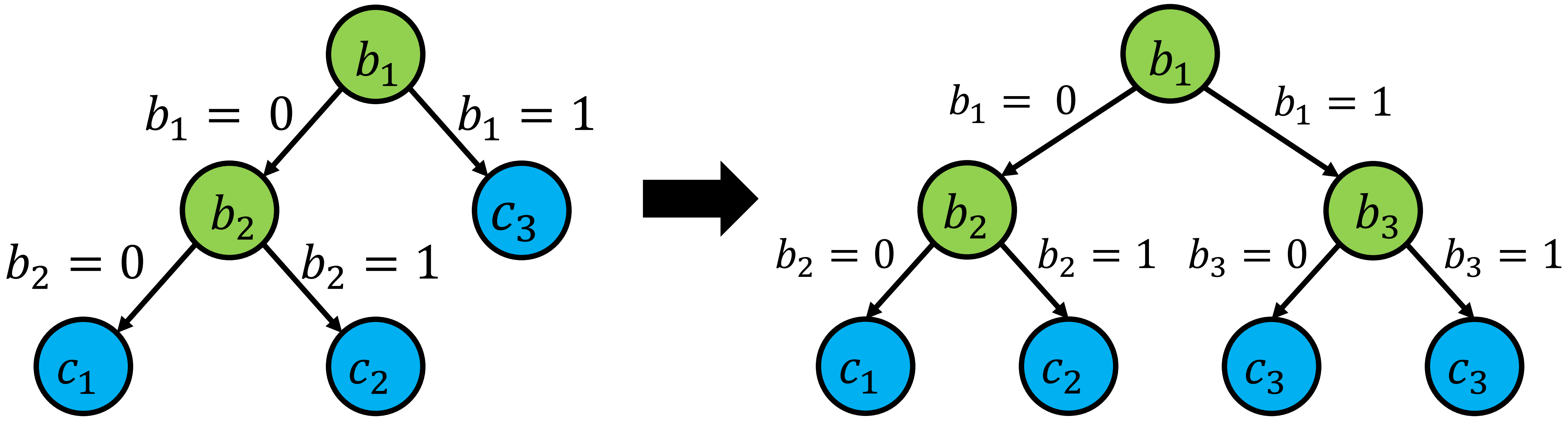


leaves become OT database



Hiding the Structure

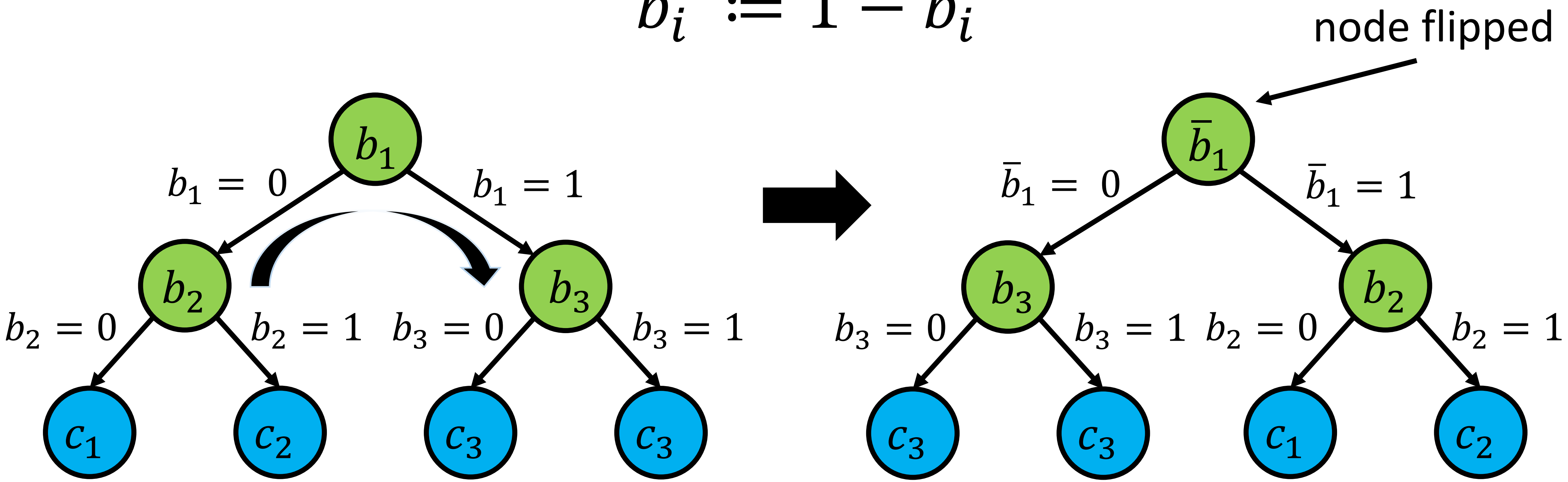
1. Padding: Insert “dummy” nodes to obtain complete tree



Hiding the Structure

2. Randomization: Randomly flip decision variables:

$$\bar{b}_i := 1 - b_i$$



Hiding the Structure: Randomization

Choose

$$s = s_1 s_2 \dots s_m \leftarrow \{0,1\}^m$$

uniformly at random

If $s_i = 1$ then flip

$$b_i \mapsto 1 - b_i$$

Semi-honest Secure Protocol

1. **Server:** Pad and randomize the decision tree
2. **Server & Client:** Engage in comparison protocol to compute each b_i
3. **Client:** Compute the index j of the leaf node
4. **Client & Server:** Engage in OT to obtain c_j

Theorem. This protocol is secure against *semi-honest* adversaries.

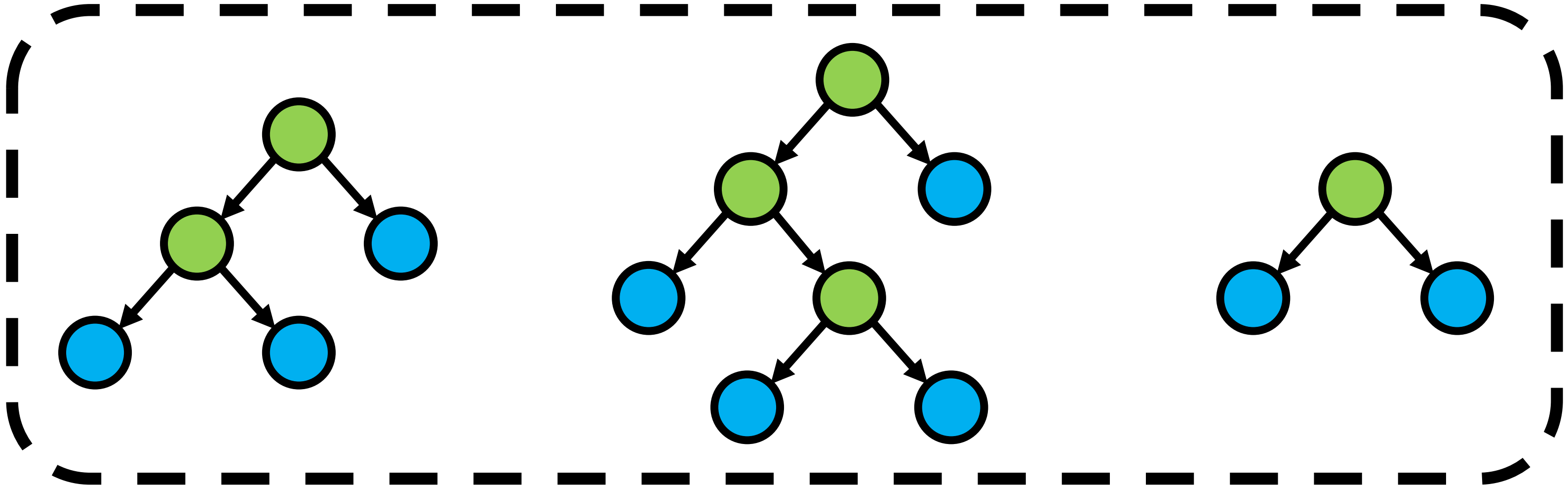
From Trees to Forests

Naïve Solution: Evaluate each tree independently using the protocol

Problem: Reveals more information about the model than just the classification

From Trees to Forests

Better Solution: Use an additive secret-sharing to hide intermediate results



Evaluate each tree as before, but each individual evaluation now looks random

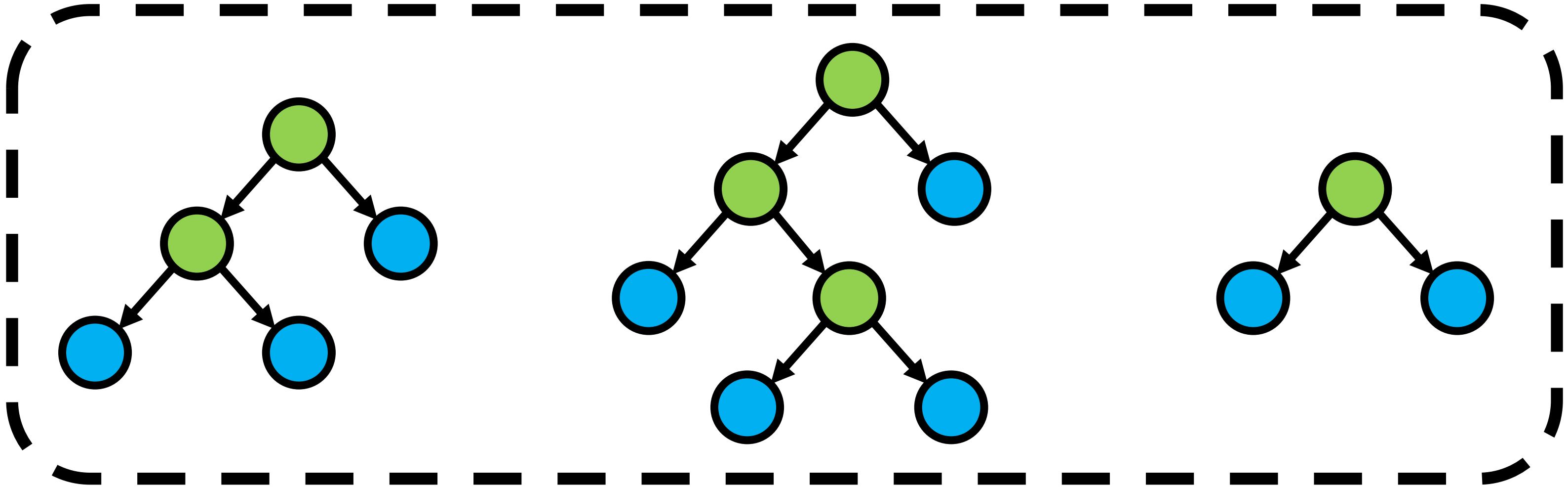
add r_1 to each classification

add r_2 to each classification

add r_3 to each classification

From Trees to Forests

Better Solution: Use an additive secret-sharing to hide intermediate results



add r_1 to each classification

add r_2 to each classification

add r_3 to each classification

Reveal $\sum_i r_i$ to the client, which allows client to learn sum (mean) of predicted values

Implementation

Implementation

Implemented private decision tree + random forest protocol (semi-honest security)

Two primary components:

- Comparison protocol
- Oblivious Transfer

Implementation

Comparison protocol instantiated with exponential variant of ElGamal encryption

- Fast instantiation using elliptic curves

Oblivious transfer based on Naor-Pinkas with OT Extensions

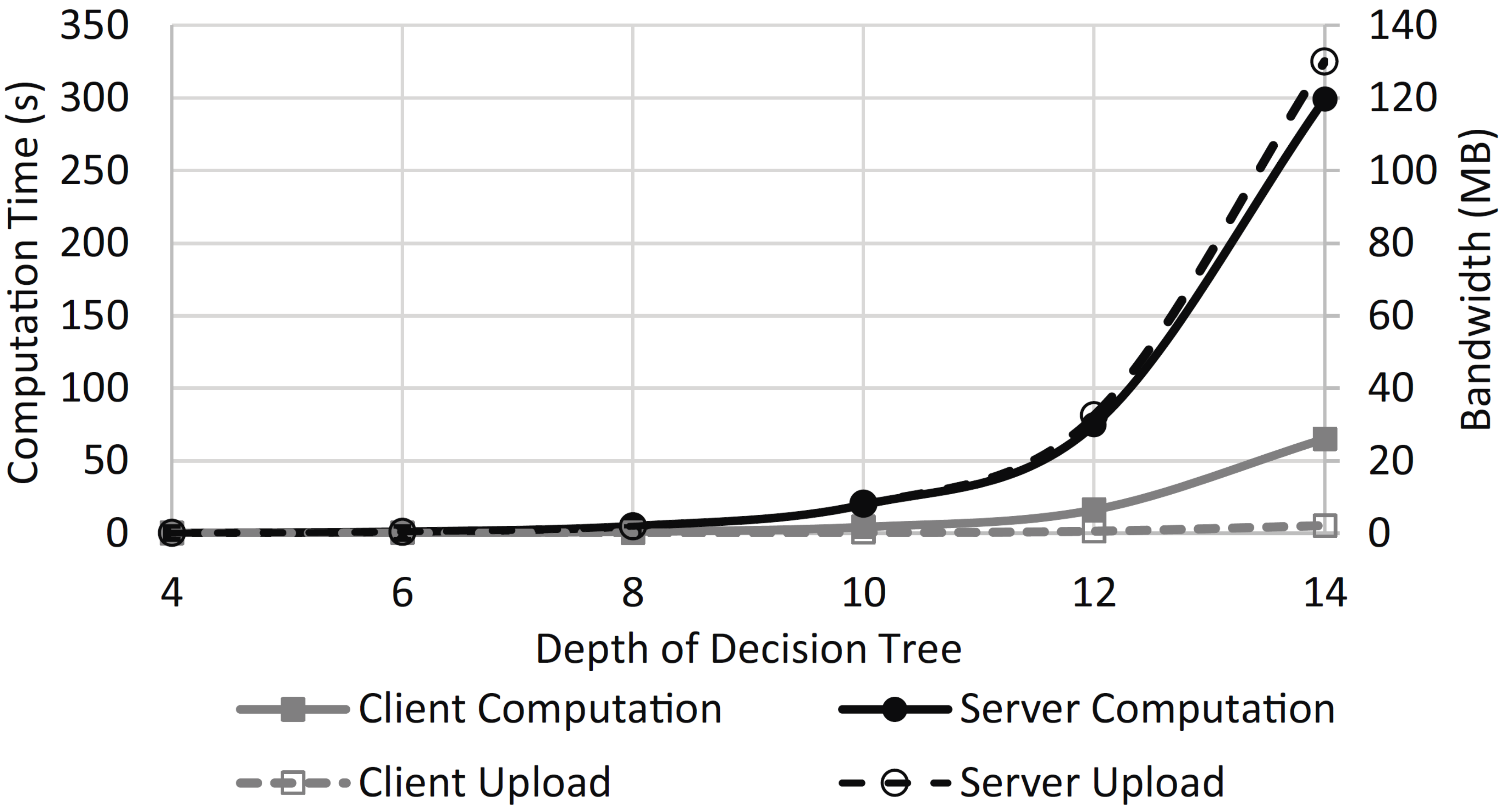
Decision Tree Evaluation on ECG Data

	Security Level	Computation (s)		Bandwidth (KB)
		Client	Server	
[BFK+09]	80	1.765	4.235	112.2
[BPGT14]	80	1.485	2.595	4272
This work	**128**	**0.091**	**0.188**	**101.9**

Experimental Parameters:

- Data Dimension: 6
- Depth of Decision Tree: 4
- Number of Comparisons: 6

Performance for Complete Decision Trees



One-Sided Security (Malicious Model)

Privacy of the server's model is ensured against a malicious client

Privacy of the client's input is ensured against a malicious server

However, client not guaranteed to receive "correct" answer

Extensions to One-Sided Security

Possible attacks on semi-honest protocol:

1. **Server:** Pad and randomize the decision tree
2. **Server & Client:** Engage in comparison protocol to compute each b_i
3. **Client:** Compute the index j of the leaf node containing the response
4. **Client & Server:** Engage in OT to obtain c_j

Client might cheat during comparison protocol (for example, encrypt a value that is not 0/1)

Solution: zero-knowledge proofs

Client might cheat by requesting a different index

Solution: “conditional” oblivious transfer

Conclusion

Simple protocols for decision tree evaluation in both semi-honest and malicious setting

Semi-honest decision tree / random forest evaluation protocols are fairly practical