Two-Party Decision Tree Training from Updatable Order-Revealing Encryption

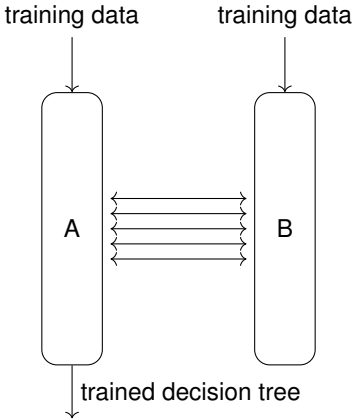
Robin Berger[1], Felix Dörre[1], Alexander Koch[2]

2024-03-07 @ACNS 2024

[1]: KASTEL, KIT

[2]: CNRS & IRIF

Our Setting



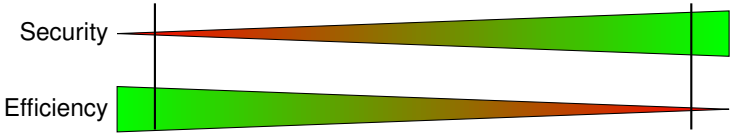
Theory/Practice

Practice:

- Focus on efficiency
- Computation on plaintext data

Theory:

- Focus on security
- Computation using generic Multiparty Computation



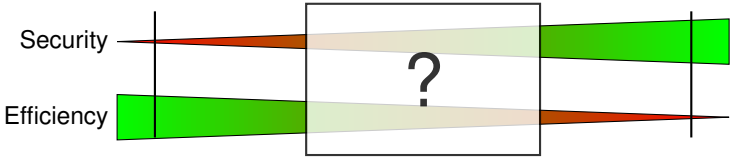
Theory/Practice

Practice:

- Focus on efficiency
- Computation on plaintext data

Theory:

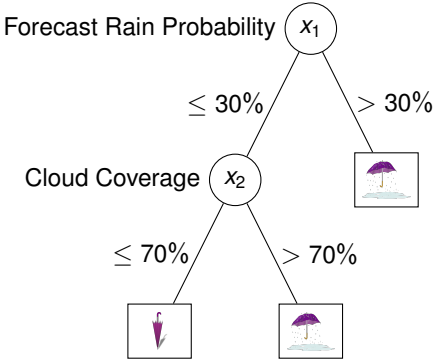
- Focus on security
- Computation using generic Multiparty Computation



Can we achieve a speedup by allowing some leakage?
(Specifically for decision tree training)

Decision Trees

Should I bring an umbrella?

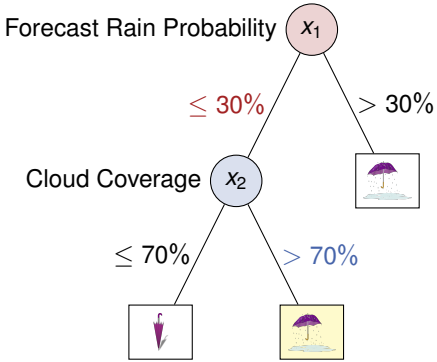


Example datapoint:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 27\% \\ 91\% \end{pmatrix}$$

Decision Trees

Should I bring an umbrella?



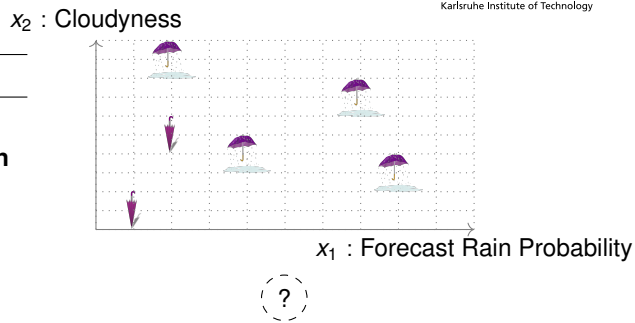
Example datapoint:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 27\% \\ 91\% \end{pmatrix}$$

How to Train a Decision Tree?

Decision Tree Training

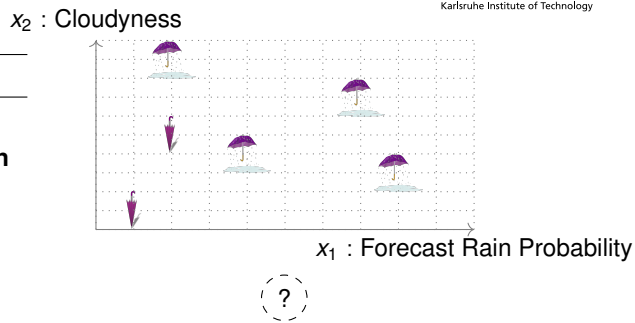
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

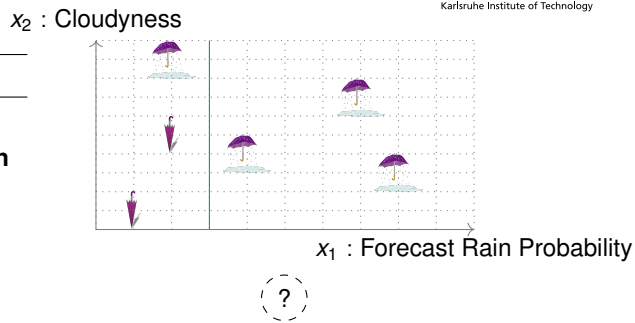
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

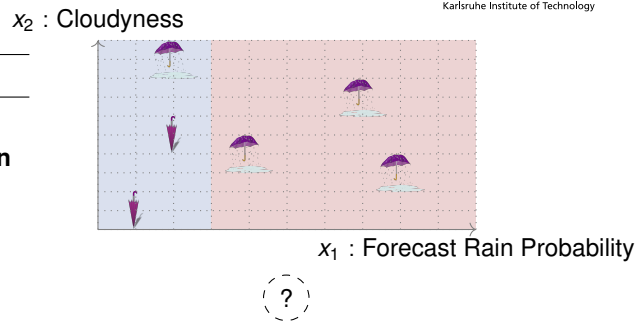
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

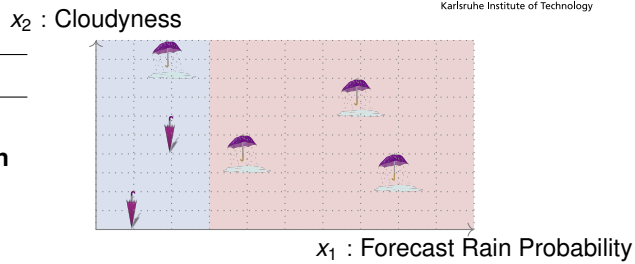
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```

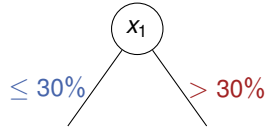


How to Train a Decision Tree?

Decision Tree Training

```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```

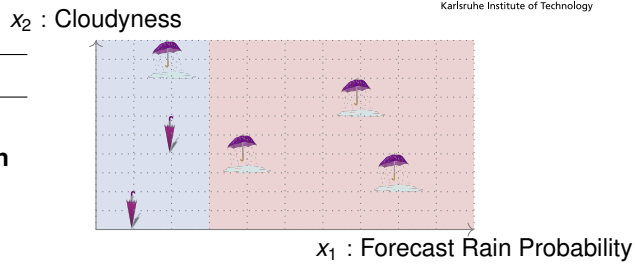


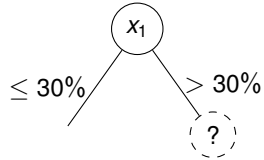


How to Train a Decision Tree?

Decision Tree Training

```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```

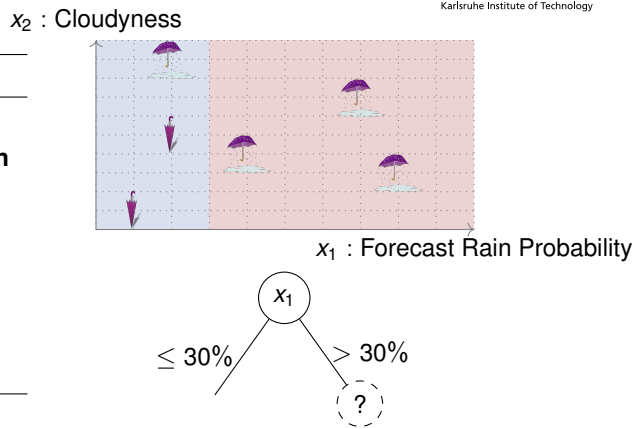




How to Train a Decision Tree?

Decision Tree Training

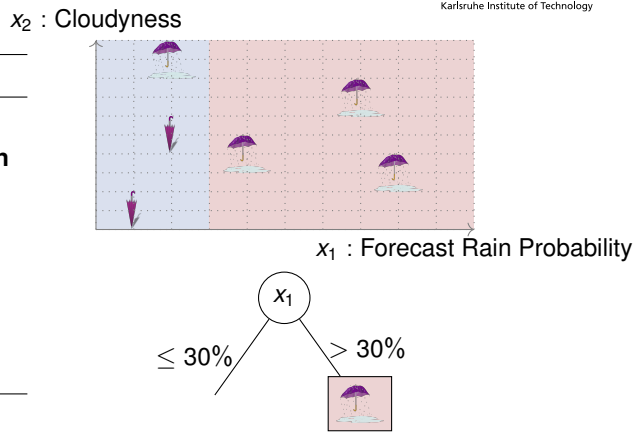
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

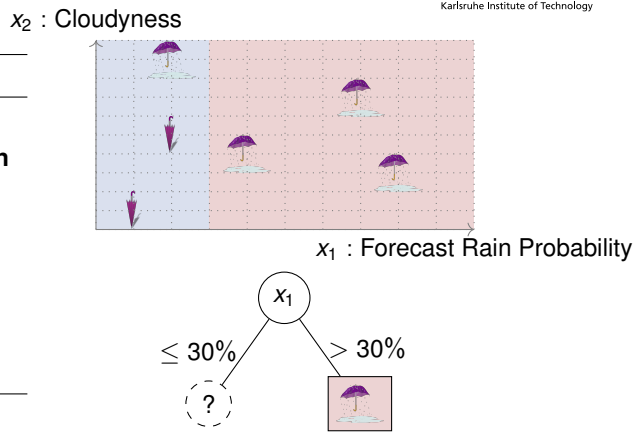
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

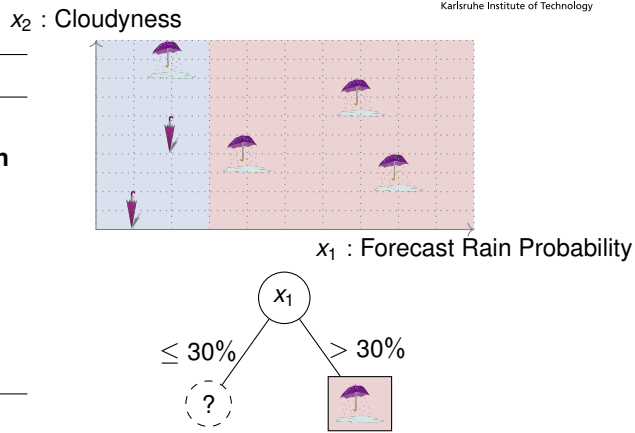
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

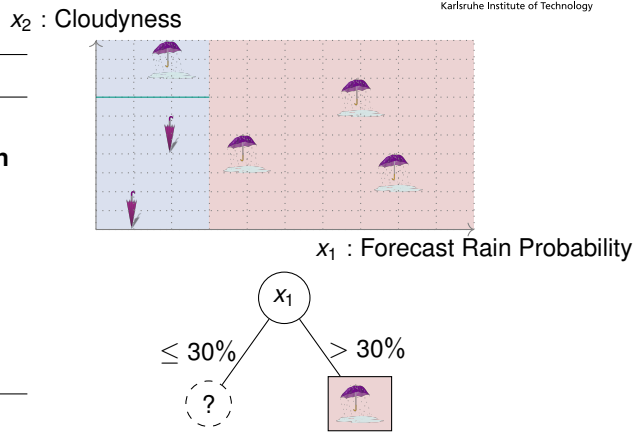
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

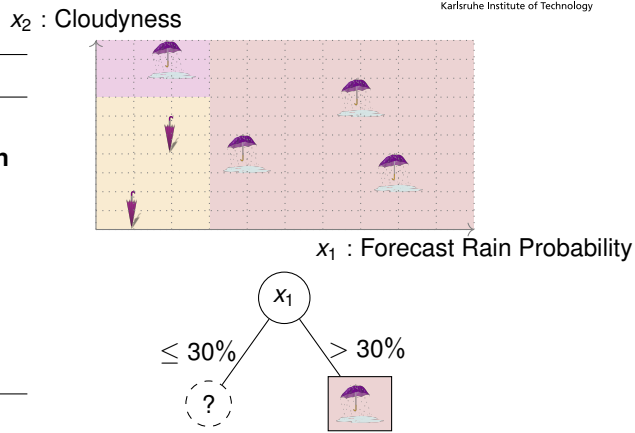
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

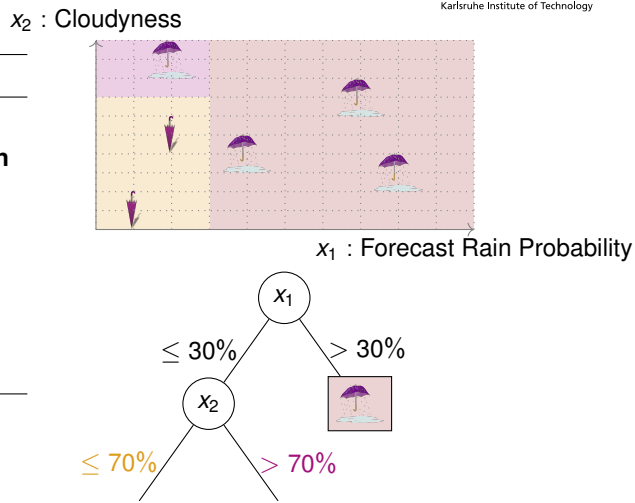
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

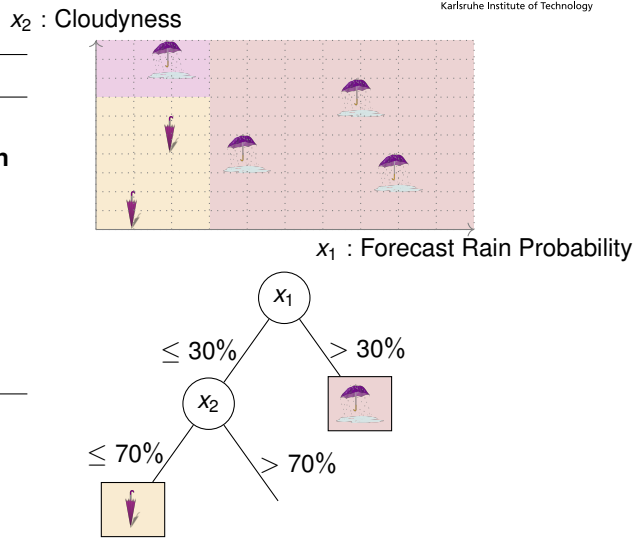
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

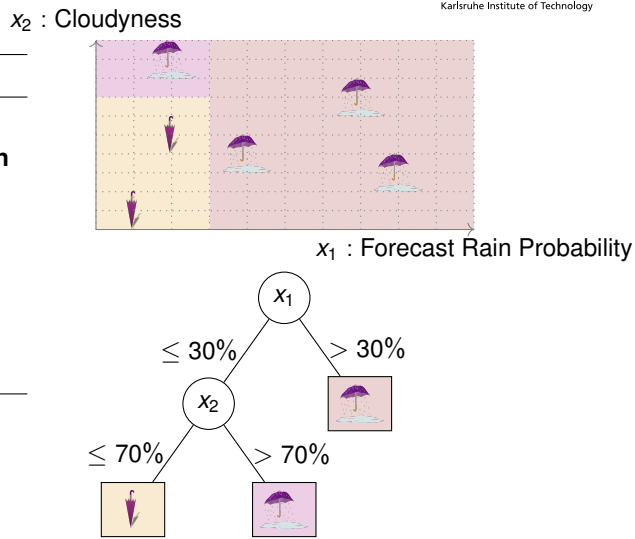
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

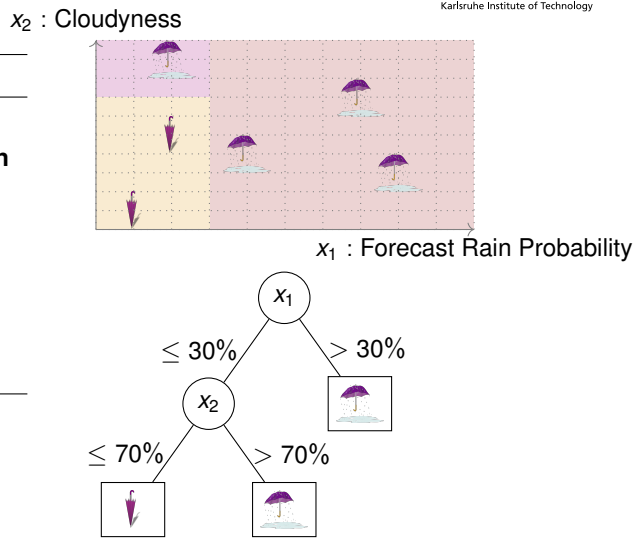
```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



How to Train a Decision Tree?

Decision Tree Training

```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```



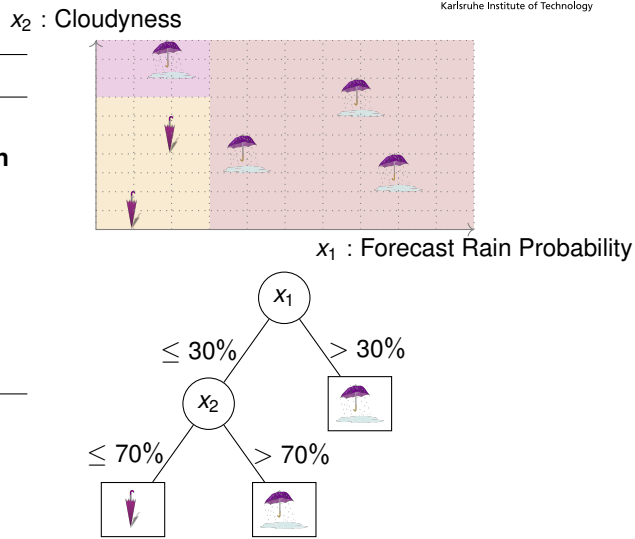
How to Train a Decision Tree?

Decision Tree Training

```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```

Required operations:

- Equality checks
- Comparisons



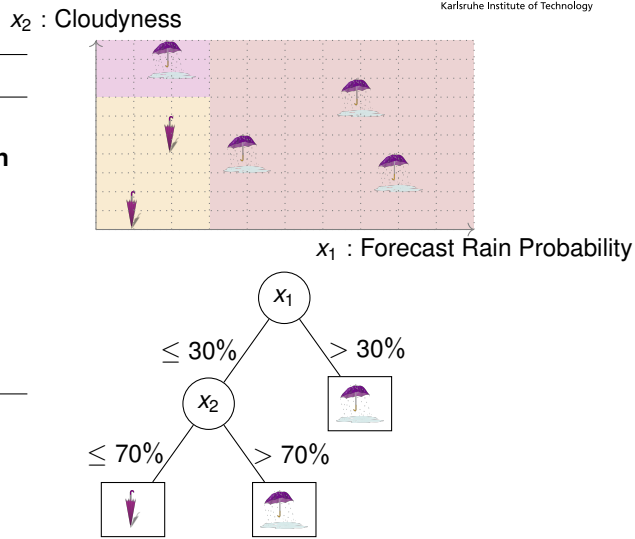
How to Train a Decision Tree?

Decision Tree Training

```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```

Required operations:

- Equality checks
- Comparisons



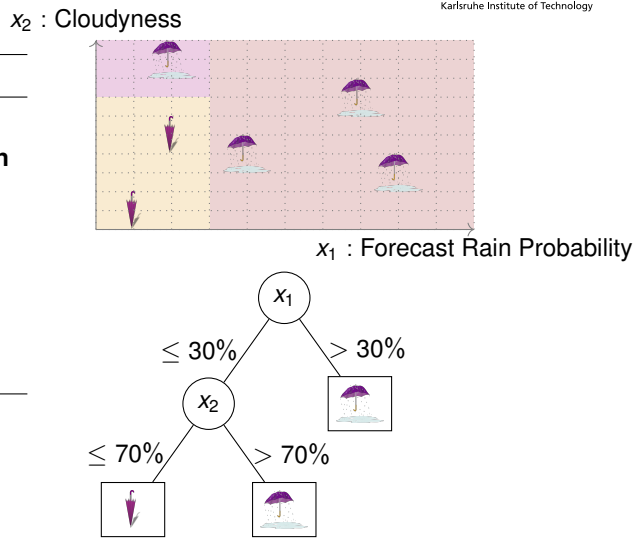
How to Train a Decision Tree?

Decision Tree Training

```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```

Required operations:

- Equality checks
- Comparisons



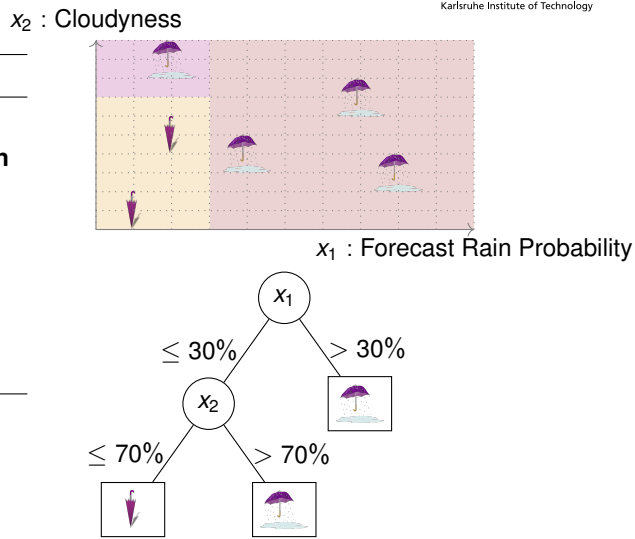
How to Train a Decision Tree?

Decision Tree Training

```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
    end if
    split = FINDBESTSPLIT(data)
    (X, Y) := split(data)
    return InnerNode(split, TRAIN(X), TRAIN(Y))
end function
```

Required operations:

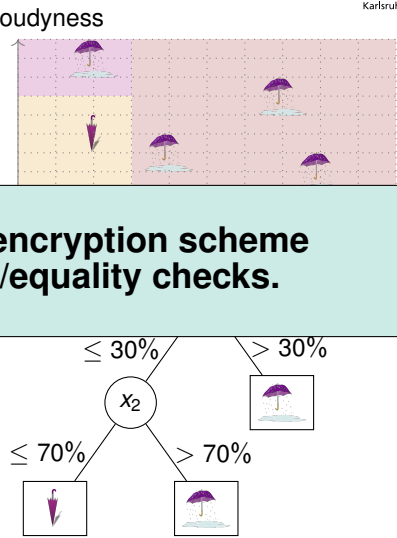
- Equality checks
- Comparisons



How to Train a Decision Tree?

Decision Tree Training

```
function TRAIN(data)
    if all datapoints in data have the same label then
        return Leaf(label)
end function
```



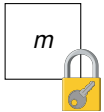
Idea: Encrypt data with an encryption scheme that permits comparisons/equality checks.

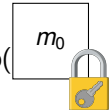
Required operations:

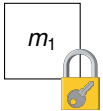
- Equality checks
- Comparisons

Order-Revealing Encryption

An Order-Revealing Encryption scheme consists of the following algorithms:

- Gen $\rightarrow$ 
- Enc( , m) $\rightarrow$ 

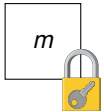
- Cmp(

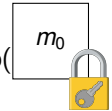
, 

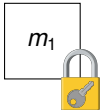
) $\rightarrow m_0 \overset{?}{<} m_1$

Order-Revealing Encryption

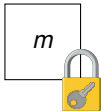
An Order-Revealing Encryption scheme consists of the following algorithms:

- Gen $\rightarrow$ 
- Enc( , m) $\rightarrow$ 

- Cmp(

, 

) $\rightarrow m_0 \overset{?}{<} m_1$

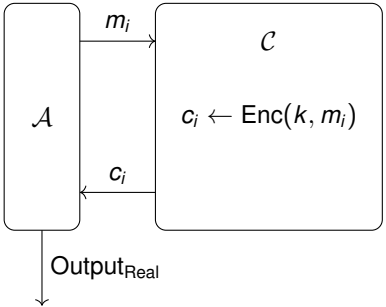
$*$ Dec( , 

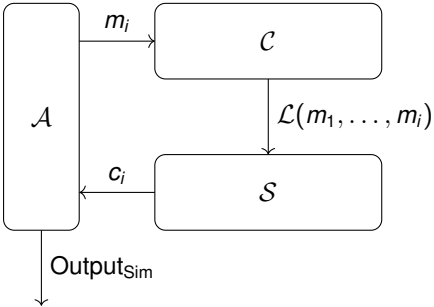
) $\rightarrow m$

Security of Order-Revealing Encryption

ORE security w.r.t. leakage $\mathcal{L}$ from [Che+16]

There exists a PPT $\mathcal{S}$, such that for all PPT $\mathcal{A}$: $\text{Output}_{\text{Real}} \overset{c}{\approx} \text{Output}_{\text{Sim}}$

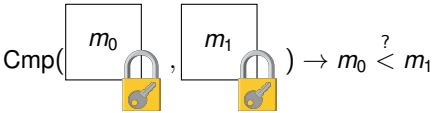


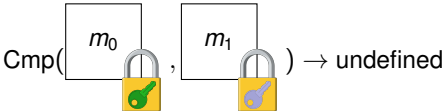


Comparisons on Ciphertexts

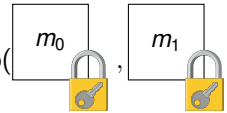
$\mathrm{Cmp}(\boxed{m_0}\,\text{🔒}, \boxed{m_1}\,\text{🔒}) \rightarrow m_0 \overset{?}{<} m_1$

Comparisons on Ciphertexts

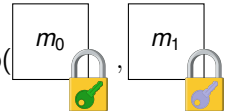




Comparisons on Ciphertexts

Cmp(

) $\rightarrow m_0 \overset{?}{<} m_1$

Cmp(

) $\rightarrow$ undefined

Problem when applied to decision tree training:

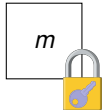
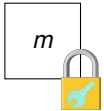
- All parties need to use the same  .

Updatable Order-Revealing Encryption

Updatable Order-Revealing Encryption adds the following algorithms:

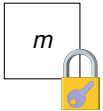
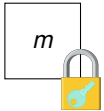
Updatable Order-Revealing Encryption

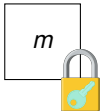
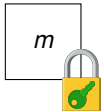
Updatable Order-Revealing Encryption adds the following algorithms:

- Upd( , ) → 

Updatable Order-Revealing Encryption

Updatable Order-Revealing Encryption adds the following algorithms:

- Upd( , ) → 

- $\mathrm{Upd}^{-1}($  , ) → 

Interactive Encryption

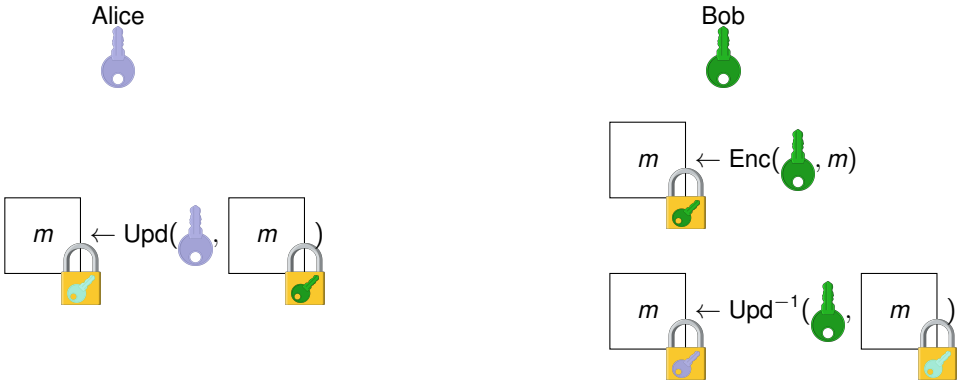
Alice

Bob

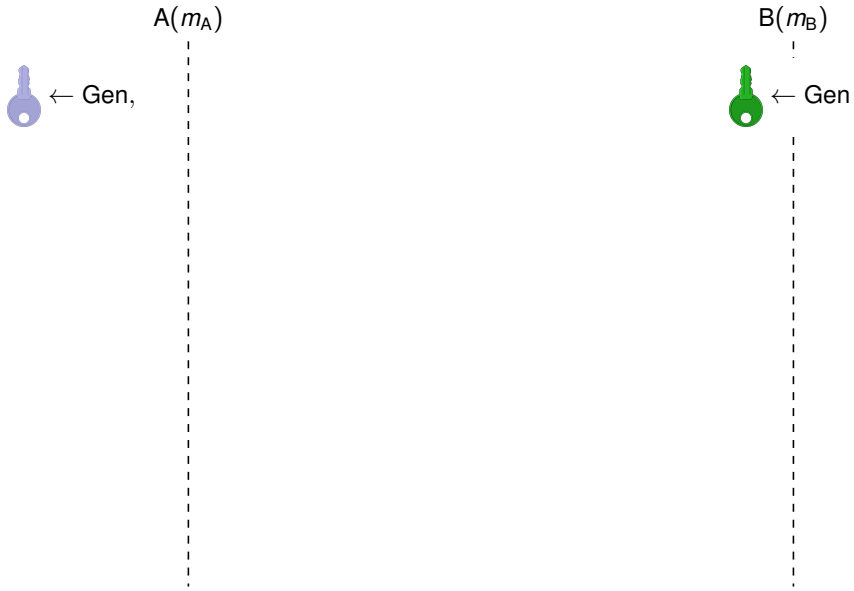
Interactive Encryption

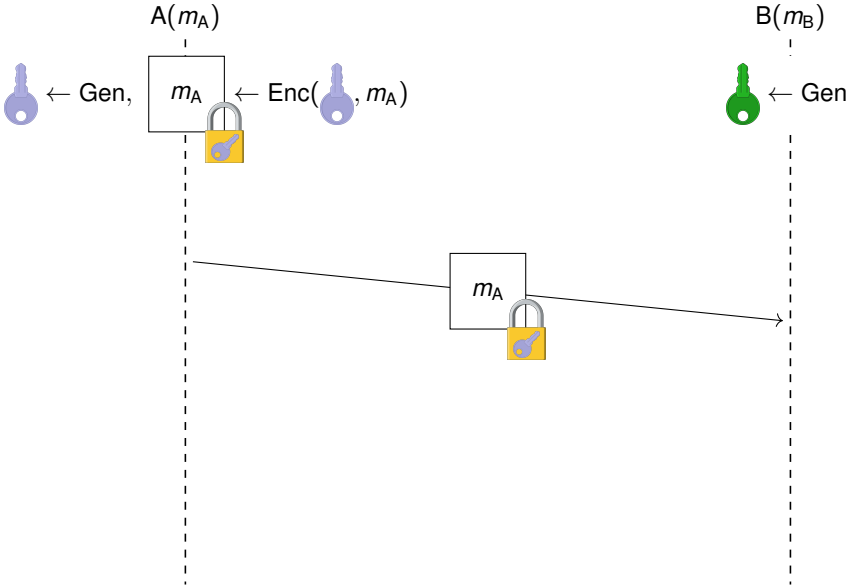
Interactive Encryption

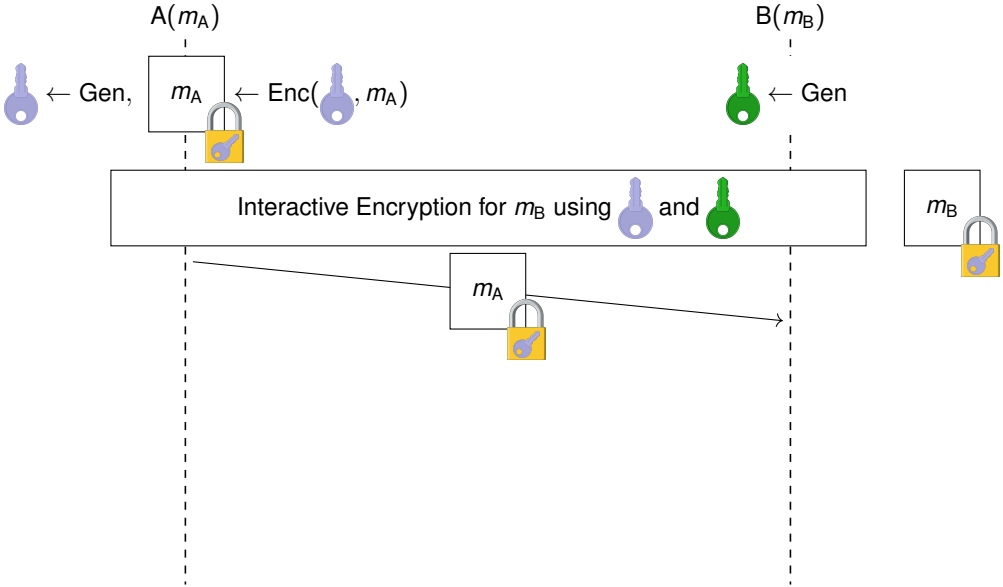
Interactive Encryption

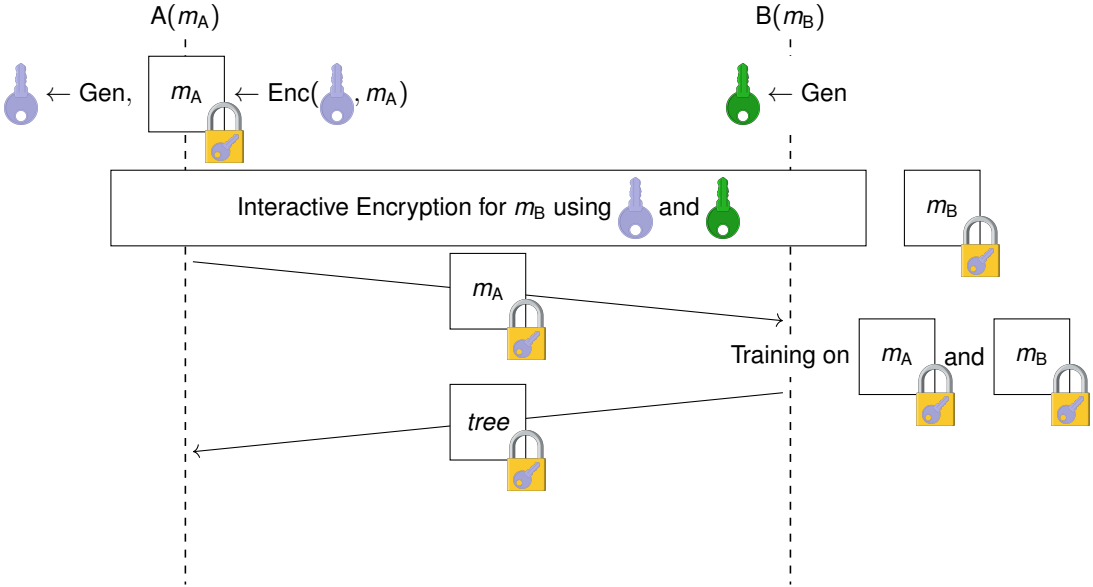


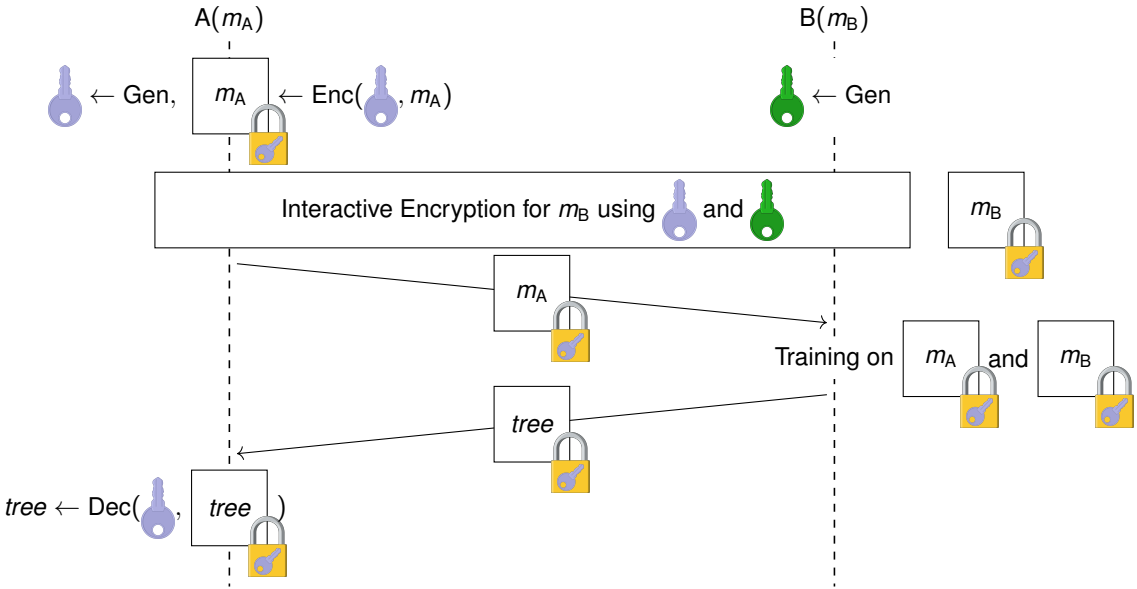
A(m_A) B(m_B)


A(m_A)
B(m_B)
← Gen,
← Gen

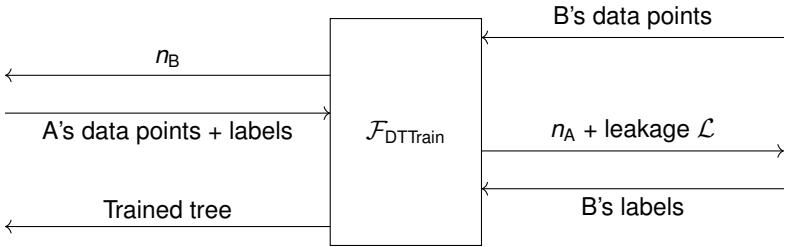

A(m_A)
B(m_B)
← Gen,
m_A
← Enc(, m_A)
← Gen
m_A


A(m_A)
B(m_B)
← Gen,
m_A
← Enc(, m_A)
← Gen
Interactive Encryption for m_B using and
m_B
m_A


A(m_A)
B(m_B)
← Gen,
m_A
← Enc(, m_A)
← Gen
Interactive Encryption for m_B using and
m_B
m_A
Training on m_A and m_B
tree


A(m_A)
B(m_B)
← Gen,
m_A
← Enc(, m_A)
← Gen
Interactive Encryption for m_B using and
m_B
m_A
Training on m_A and m_B
tree
tree ← Dec(, tree)

Semi-Honest Security of the Protocol



Performance evaluation

Benchmarks on

- Synthetic dataset
  - 2^{13} samples
  - 11 attributes
- 50ms network latency

Training Protocol	Runtime
Plaintext training	0.4s
Our work	20.1s
Hamada et al. [Ham+23]	4821.6s

Conclusion

Efficient approach for decision tree training if

- Training algorithm only requires comparisons and equality checks
- Leakage is acceptable
  - Case-by-case decision

References

[Che+16] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. “Practical Order-Revealing Encryption with Limited Leakage”. In: 2016, pp. 474–493. DOI: 10.1007/978-3-662-52993-5_24.

[Ham+23] Koki Hamada, Dai Ikarashi, Ryo Kikuchi, and Koji Chida. “Efficient decision tree training with new data structure for secure multi-party computation”. In: 2023.1 (Jan. 2023), pp. 343–364. DOI: 10.56553/popets-2023-0021.

Leakage

Leakage of our scheme:

$$\mathcal{L}(m_1, \ldots, m_N) = \{(i, j, \mathrm{hsb}(m_i \oplus m_j)) \mid m_i < m_j\}$$

Example:

message	leakage
00100	00???
01110	01???
10000	1??0?
10011	1==1?

Maximum leakage: $\mathcal{O}(\log N)$ bits per message