Decision Tree Evaluation via HE

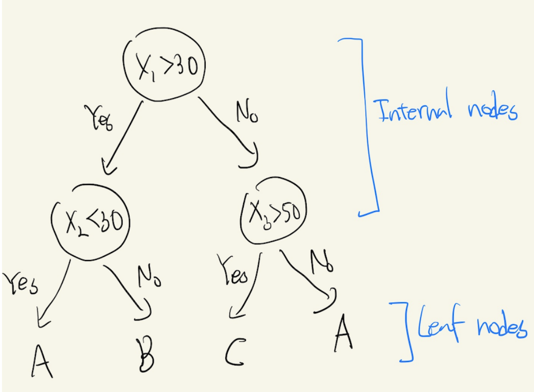
Sunghyeon Jo

Seoul National University

Cryptography & Privacy Lab
Nov 7, 2023

Overview

Decision Tree



If input $X = [X_1, X_2, X_3] = [40, 50, 30]$, then evalutation result of the decision tree is B.

Classifying Private Data with Decision Trees

Scenario: The client possesses private data and the server owns a decision tree.

Challenge:

- Naive Evaluation: Executes in $O(d)$ time, where d is the depth of the decision tree. This approach reveals data.
- Goal: Classify the client's private data without exposing it to the server.

Idea:

- Client sends data encrypted using homomorphic encryption.
- Server evaluates encrypted data without ever decrypting it.

Papers Under Discussion

1. SortingHat: Efficient Private Decision Tree Evaluation via Homomorphic Encryption and Transciphering

- *Published in*: CCS 2022
- *Method*: Fully Homomorphic Encryption (FHE)
- *Implementation*: TFHE

2. Level Up: Private Non-Interactive Decision Tree Evaluation using Levelled Homomorphic Encryption

- *Published in*: CCS 2023
- *Method*: Levelled Homomorphic Encryption (Leveled HE)
- *Implementation*: BFV

SortingHat Overview

- **Homomorphic Comparison, Homomorphic Traversal**: Reducing execution time - mostly homomorphic multiplication
- **Transciphering**: a General Strategy to Reduce Communication Cost

Homomorphic Comparison

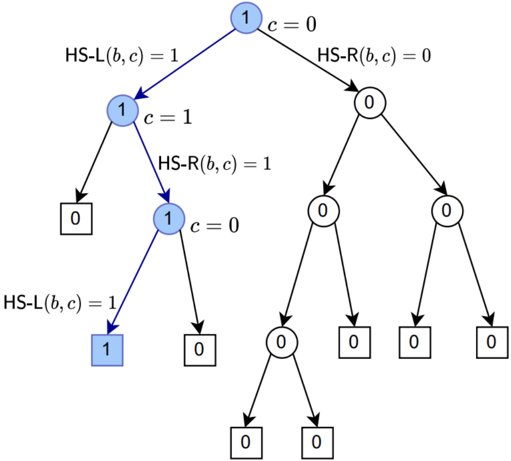
Algorithm: Comparison function PolyComp

Require: $c := RLWE(X^M)$ and a threshold value t
Ensure: $c' = RLWE(M(X))$; $M_0 = 1$ if $M \geq t$, otherwise $M_0 = 0$, where M_0 is the constant term of $M(X)$

1: Let $T(X) := X^{2N-N} + X^{2N-(N-1)} + \cdots + X^{2N-t}$
2: compute $Plain.Mult(c, T(X)) \rightarrow c'$
3: **return** c'

Downside: threshold value t's bitlength limited to $\log N$

Homomorphic Traversal



Homomorphic Traversal

Traversal of decision tree and finding final leaf node

- HS-R$(b, c) := c \cdot b$, HS-L$(b, c) := (1 - c) \cdot b$
- using HS-R(b, c) value, HS-L(b, c) can be evaluated without homomorphic Multiplication.
- If the comparison result is saved in the form of an RGSW ciphertext, we can process homomorphic traversal via calculating external products.

Homomorphic Traversal

Algorithm 8 Server's computation for PDTE.

Input: $\{c_{i,j}\}_{i\in[0,\ldots,n-1],j\in[0,\ldots,\ell-1]}$, ksk and classification labels $\tau_0,\ldots,\tau_{k-1}$
Output: $c := \mathsf{RLWE}_{N,t,q}(\mathcal{T}(x_0,\ldots,x_{n-1}))$ which is the resulting classification label
$c \leftarrow 0$
$a \leftarrow 1$
for $i \leftarrow 0\ldots m-1$ **do**
  for $j \leftarrow 0\ldots \ell-1$ **do**
    Let $\mathbf{t}(i)$ be the threshold value of i-th node.
    Run $\mathsf{PolyComp}(c_{\mathbf{a}(i),j}, \mathbf{t}(i)) \rightarrow b_{i,j}$
  end for
  Run $\mathsf{RLWEtoRGSW}(\{b_{i,j})\}_{j\in[0,\ldots,\ell-1]}, \mathsf{ksk}) \rightarrow \hat{c}_i$
end for
Run $\mathsf{HomTrav}(\{\hat{c}_i\}_{i\in[0,\ldots,m-1]}, a) \rightarrow \{z_l\}_{l\in\mathcal{L}}$
for $l \in \mathcal{L}$ **do**
  $c \leftarrow c + \mathsf{Plain.Mult}(z_l, \mathsf{lab}(l))$
end for
return c.

Transciphering

- FHE schemes have longer ciphertexts $\rightarrow$ bigger communication overheads
- To improve efficiency, encrypt data with a symmetric cipher and then decrypt it homomorphically on the server.
    - Messages are sent from client to server using a symmetric key cipher
    - Homomorphically decrypt these messages using the encrypted secret keys
- FiLIP cipher is designed for such FHE-friendly operations.

FiLIP cipher

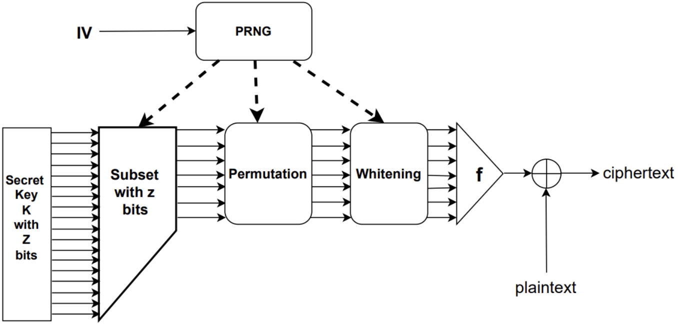


Fig. 1: FiLIP encryption of one bit. Using the PRNG, we select a subset of the bits of the keys, then we shuffle them and apply an XOR with each bit of the whitening vector. Finally, we apply the non-linear function f and XOR the result with the plaintext.

Transciphering

- transciphering takes as input a ciphertext $c = \text{FiLIP}.enc(m)$ and outputs a low-noise LWE encryption c' of m.
- a setup phase where the client sends to the server RLWE and RGSW encryptions of each bit k_i of FiLIP's secret key K
- Limitations
    - The transciphering method with the FiLIP cipher only encrypts single bits (with plaintext modulus $t = 2$)
    - The output ciphertexts are limited to binary values 0, 1.
    - Homomorphic bit operations are necessary for homomorphic traversal (requires multiple external multiplications)

Homomorphic Comparison for t-sortingHat

Consider μ-bit integers m, v and its representation $m_0, \cdots m_{\mu-1}$ and $v_0, \cdots v_{\mu-1}$. ($m = \sum_{i=0}^{\mu-1} 2^i m_i$)

Assume that we know $Enc(m_0), \cdots, Enc(m_{\mu-1})$ amd $v_0, \cdots, v_{\mu-1}$

$m > v$ means $m_{\mu-1} = v_{\mu-1}, \cdots, m_{k+1} = v_{k+1}, m_k > v_k$ holds for some k.

Homomorphic Comparison for t-sortingHat

Let $X_k(v) = 1$ if $m_{\mu-1} = v_{\mu-1}, \cdots, m_k = v_k$, otherwise 0.

Then $X_k(v) = \text{XNOR}(Enc(m_k), v_k) \cdots \text{XNOR}(Enc(m_{\mu-1}), v_{\mu-1})$
where $\text{XNOR}(Enc(a), b)$ for $a, b \in \{0, 1\}$ returns 1 iff $a = b$

Let $R_k(v) := m_{k\ldots\mu-1} > v_{k\ldots\mu-1}$
Then, $R_k(v) =$
$\text{Enc}(m_{\mu-1}) \cdot v_{\mu-1} + X_{\mu-1}(v) \cdot \text{Enc}(m_{\mu-2}) \cdot v_{\mu-2} + \cdots + X_{k+1}(v) \cdot \text{Enc}(m_k) \cdot v_k$

Homomorphic Comparison for t-sortingHat

Algorithm XNOR

Require: $Enc(m_i)$ and v_i, where $m_i, v_i \in \{0, 1\}$

Ensure: $Enc(b)$ such that $b = 1$ if $m_i = v_i$ and $b = 0$ otherwise

```
if v_i = 1 then
    return Enc(m_i)
else
    return NOT(Enc(m_i))
end if
```

Homomorphic Comparison for t-sortingHat

Want to do: Given $Enc(m_0), \cdots, Enc(m_{\mu-1})$, Compare m and n known values $v^{(1)}, \cdots, v^{(n)}$.

- Using dynamic programming
  1. for $k = \mu - 1, \mu - 1, \cdots 0$, execute following steps
  2. calculate $X(v^{(i)}), R(v^{(i)})$ for all distinct value $v^{(i)}_{k..\mu-1}$.
- $\min(O(n \cdot \mu), O(2^\mu))$ homomorphic multiplication required
- Using divide and conquer method (halves μ bits to high $\mu/2$ and low $\mu/2$) and do recursively until $2^\mu < n$, and merge the results R and X.
  - divide and conquer algorithm require $O(n \cdot \mu / \log m)$ homomorphic multiplications

SortingHat

Homomorphic AND gate consists of $\bar{n}$ external products

	[TBK20]	SortingHat	t-SortingHat
Comp	$O(m \cdot \bar{n} \cdot \mu \cdot \log \mu \cdot \log q)$	$O(m)$	$O(\frac{m \cdot \mu}{\log m})$
EvalTree	$2 \cdot m \cdot \bar{n}$	$m \cdot (\log N + 1)$	$m \cdot \bar{n}$

LevelUp

- FHE: unlimited multiplicative depth, but slow and inefficient.
- Use leveled HE and keep multiplicative depth as small as possible
- Multiplicative depth is depend on decision tree's depth

LevelUp

CCS '23, November 26–30, 2023, Copenhagen, Denmark Mahdavi et al.

	PROBONITE [8]	PDT-Bin [39]	PDT-Int [39]	SortingHats [15]	XXCMP-PDTE	RCC-PDTE
Supports Unbalanced	×	✓	✓	✓	✓	✓
Attribute Selection	PIR	Clear	Clear	Clear	Clear	Clear
Comparison	PBS (CT-CT)	Folklore	Lin-Tzeng [26]	XCMP-CT-PT	**XXCMP**	**RCC**
Path Evaluation	CMux	AND	SumPath	CMux	SumPath	SumPath
Batchable	×	✓	✓	×	✓	✓
Bit Precision	$< 8^*$	n	n	11	n	n
Levelled or FHE	FHE	LHE or FHE	LHE(BGV)	FHE(TFHE)	LHE(FV)	LHE(FV)
# of Comparisons	$O(d)$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}\rvert$
Query Complexity	$O(\lvert\mathbf{x}\rvert)$	$O(n\lvert\mathbf{x}\rvert)$	$O(n\lvert\mathbf{x}\rvert)$	$O(N\lvert\mathbf{x}\rvert)$	$O(N\lvert\mathbf{x}\rvert)$	$O(\ell n\lvert\mathbf{x}\rvert)$
Mult. Depth	N/A	$\log_2 n$	$\log_2 n$	N/A	$\lceil \log_2(n/\log_2 N) \rceil$	$1 + \log_2 h$

Table 3: Properties of Non-interactive Private Decision Tree Evaluation Protocols. a **is the number of client attributes,** d **is the depth of the tree,** $\mathcal{D}$ **is the set of internal decision nodes, and x is the client attribute vector.** * **The precision of PROBONITE depends on the choice of parameters for the LWE scheme but is typically less than 8 bits.**

LevelUp: XXCMP Comparison

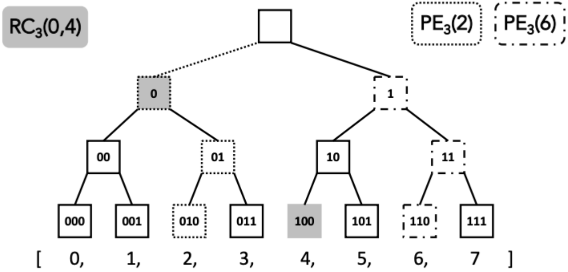
Algorithm 4 Computing $\mathbb{I}[a > b]$ using Extended XCMP (XXCMP) for $a, b \in [N^2]$ such that $a = a_1 N + a_0$ and $b = b_1 N + b_0$ where $a_i, b_i \in [N]$

procedure $\mathrm{XCMP}_0(X^a, b)$ ▷ $a, b \in [N]$
    $T \leftarrow -(1 + X + \cdots + X^{N-b-1})$
    $R \xleftarrow{\$} R_p$ and $R[0] = 0 \mod p$
    $C_0 = X^a \cdot T + R$
    return C_0

procedure $\mathrm{XXCMP}_2(A, b)$ ▷ $A \in R_p^2, b \in [N^2]$
    $X^{a_1}, X^{a_0} \leftarrow A$
    $gt_0 \leftarrow \mathrm{XCMP}_0(X^{a_0}, b_0)$
    $gt_1 \leftarrow \mathrm{XCMP}_0(X^{a_1}, b_1)$
    $eq_1 \leftarrow$ Oblivious-Expansion(X^{a_1}, b_1)
    $C = gt_1 + eq_1 \cdot gt_0$
    return C

LevelUp: Range Cover Comparison(RCC)

For $a, b \in [2^n]$, $a \leq b \Leftrightarrow b \in [a, 2^n - 1]$.



$\therefore a \leq b \Leftrightarrow \mathsf{RC}_i(a, 2^n - 1) = \mathsf{PE}_i(b)$ for some i.

LevelUp

OURC: RC$(a, 2^n - 1)$
CWENCODE: Constant-weight encoding. (length l, weight h)

Algorithm 6 OURC and PE Encoding

procedure OURC-Encode(a, h, ℓ, n) ▷ $a \in [2^n]$
  $[a_0, a_1, \cdots, a_n] \leftarrow \mathrm{OURC}(a, n)$
  for $i \in [n+1]$ **do**
    $a'_i = \mathrm{CWEncode}(a_i, h, \ell)$ ▷ $a'_i \in \mathbb{B}^\ell$
  for $i \in [\ell]$ **do**
    $pt_{OURC}[i] = \left[a'_0[i], a'_1[i], \ldots, a'_n[i]\right]$
  return pt_{OURC}

procedure PE-Encode(b, h, ℓ, n) ▷ $b \in [2^n]$
  $[b_0, b_1, \cdots, b_n] \leftarrow \mathrm{PE}(b, n)$
  for $i \in [n+1]$ **do**
    $b'_i = \mathrm{CWEncode}(b_i, h, \ell)$ ▷ $b'_i \in \mathbb{B}^\ell$
  for $i \in [\ell]$ **do**
    $pt[i] = \left[b'_0[i], b'_1[i], \ldots, b'_n[i]\right]$
  return pt_{PE}

LevelUp

Constant-weight equality operator: x, y is l-bit, and h of them are 1 (weight $= h$).

Algorithm 3 Arithmetic Constant-weight Equality Operator [30]

procedure Arith-CW-Eq-Op(x, y) ▷ $x, y \in CW(\ell, h) \cup \{0^\ell\}$

 $h' = \sum_{i \in [\ell]} x[i] \cdot y[i]$

 $e = 1/h! \cdot \prod_{i \in [h]} (h' - i)$

return e ▷ $e \in \mathbb{B}$

$e = 1$ iff $x = y$.

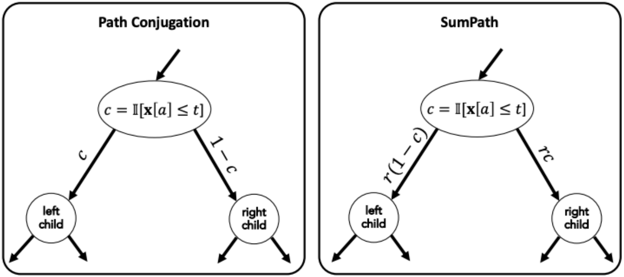
LevelUp: RCC Comparison

Algorithm 7 RCC COMPARISON

procedure RCC-COMPARE(a, b)
  $a_{ct} \leftarrow$ OURC-ENCODE(a, h, ℓ, n) ▷ Done by client
  $b_{pt} \leftarrow$ PE-ENCODE(b, h, ℓ, n)
  $\theta =$ ARITH-CW-EQ-OP(a_{ct}, b_{pt})
  $\theta_{\text{sum}} \leftarrow \sum_{i=0}^{n} \texttt{Rotate}_i(\theta)$
  $M \leftarrow 0^N, M[0] = 1$ ▷ Mask
  $\theta_{\text{cmp}} \leftarrow \theta_{\text{sum}} \otimes M$
  return θ_{cmp}

Choosing l: smallest l satisfying that $\binom{l}{h} \geq 2^n$.

Path Conjugation vs SumPath



For each leaf in the tree, the sum of all the edges in the tree from the root to that leaf is assigned to that leaf.

Only the result leaf will have 0, and all others will be non-zero.

SumPath does not require homomorphic multiplication.

LevelUp

Algorithm 8 $\boxed{\text{XXCMP-PDTE}}$: PDTE using XXCMP

procedure XXCMP-PDTE($\mathbf{x}, \mathcal{M}$)

  $(\mathcal{T}, \mathbf{a}, \mathbf{t}, \mathbf{v}) \leftarrow \mathcal{M}$

  for $d \in \mathcal{D}$ **do**

    $c \leftarrow \mathrm{XXCMP}(\mathbf{x}[\mathbf{a}[d]], \mathbf{t}[d])$

    $d.\mathrm{left} \leftarrow c$

    $d.\mathrm{right} \leftarrow 1 - c$

  for $\ell \in \mathcal{L}$ **do**

    $s(\ell)$ = Sum of edges from root to ℓ

    $r_x, r_y \xleftarrow{\$} \mathbb{Z}_p$

    $x(\ell) \leftarrow r_x \cdot s(\ell)$

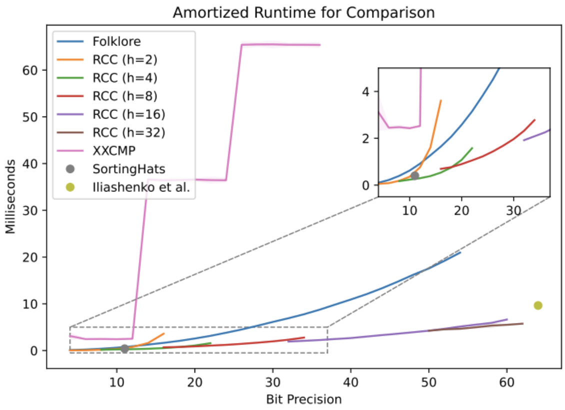
    $y(\ell) \leftarrow r_y \cdot s(\ell) + v(\ell)$

  return $\{(x(\ell), y(\ell))\}_{\ell \in \mathcal{L}}$

LevelUp: Results

- For low precision, SortingHats is superior in terms of communication, but RCC-PDTE and XXCMP-PDTE are generally faster.
- When dealing with bit precision higher than 11, XXCMP-PDTE and RCC-PDTE are the only practical available options, with RCC-PDTE proving to be faster, particularly as the number of decision nodes increases.

LevelUp


Amortized Runtime for Comparison
Folklore
RCC (h=2)
RCC (h=4)
RCC (h=8)
RCC (h=16)
RCC (h=32)
XXCMP
SortingHats
Iliashenko et al.
Milliseconds
Bit Precision
0
10
20
30
40
50
60

LevelUp

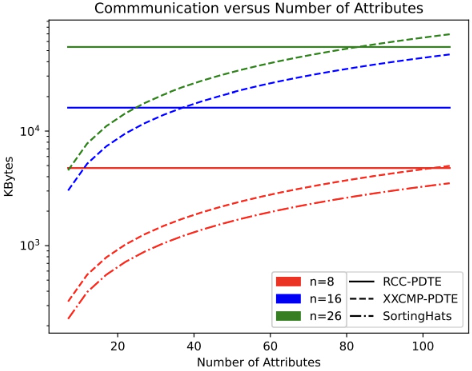


Figure 6: Communication cost of PDTE as a function of the number of attributes.