A toolbox based on homomorphic encryption for data-oblivious algorithms

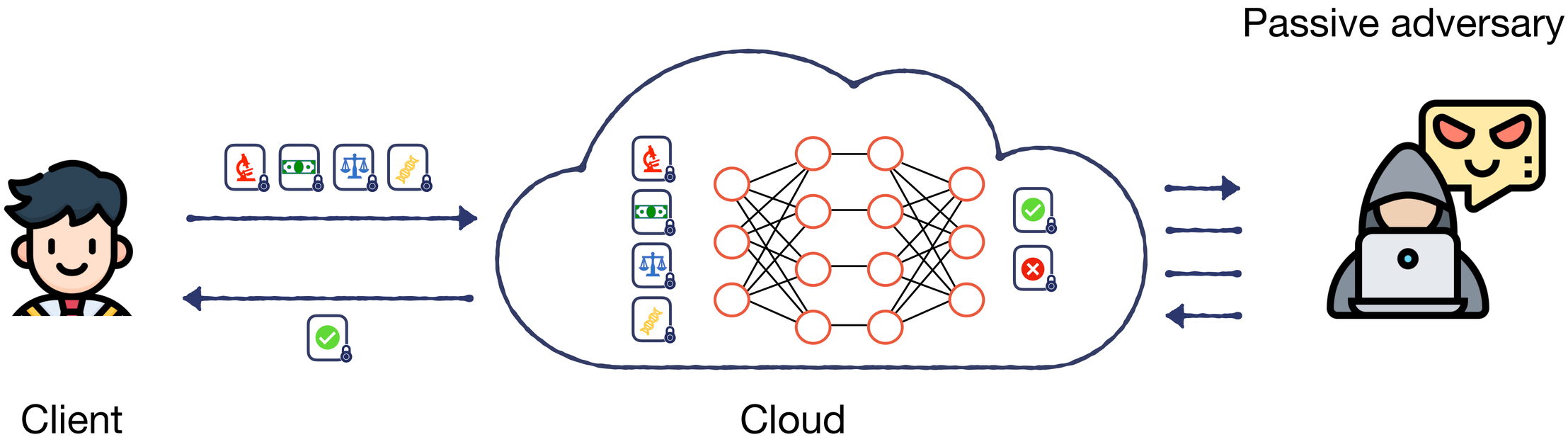
Sofiane Azogagh, Aubin Birba, Victor Delfour, Sébastien Gambs, Marc-Olivier Killijian and Felix Larose-Gervais

APVP25 - June 12th 2025

Introduction and context

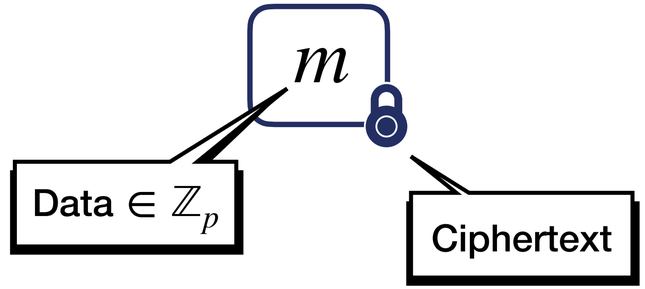
Context and security model

Outsourcing the computation



Fully Homomorphic Encryption

Fully Homomorphic Encryption



TFHE

Addition

$x + y = x + y$

Absorption

$x \times y = x \times y$

Multiplication

$x \times y = x \times y$

Arithmetic operations

Function evaluation

$f(x) = f(x)$

$f(0)$ $f(1)$ $f(2)$ ..... $f(p-1)$

LUT : Look-Up-Table

Non arithmetic operations

Chillotti, I., Gama, N., Georgieva, M. et al. TFHE: Fast Fully Homomorphic Encryption Over the Torus. J Cryptol 33, 34–91 (2020)

RevoLUT : Rust Efficient Versatile Oblivious Look-Up-Table

RevoLUT

A library for data-oblivious operations

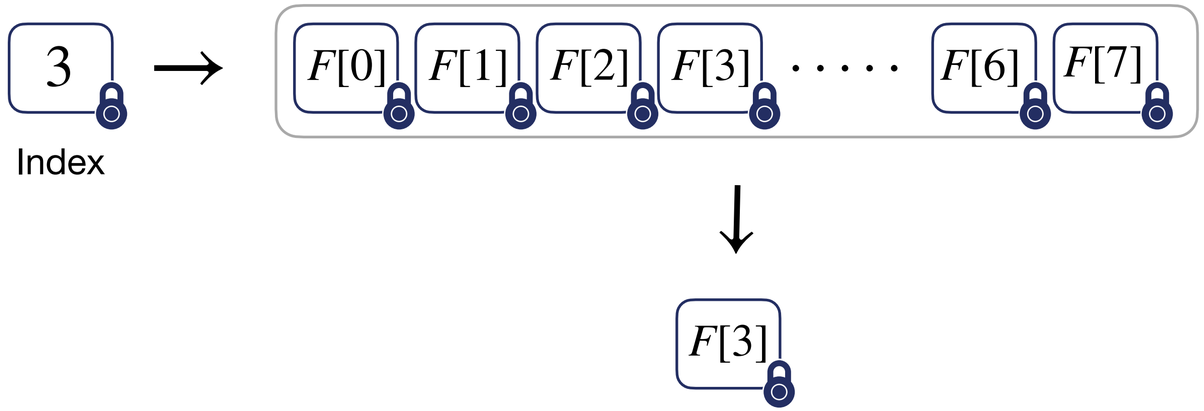
ZAMA
TFHE-rs

RevoLUT

A library for data-oblivious operations

ZAMA
TFHE-rs

Blind Array Access

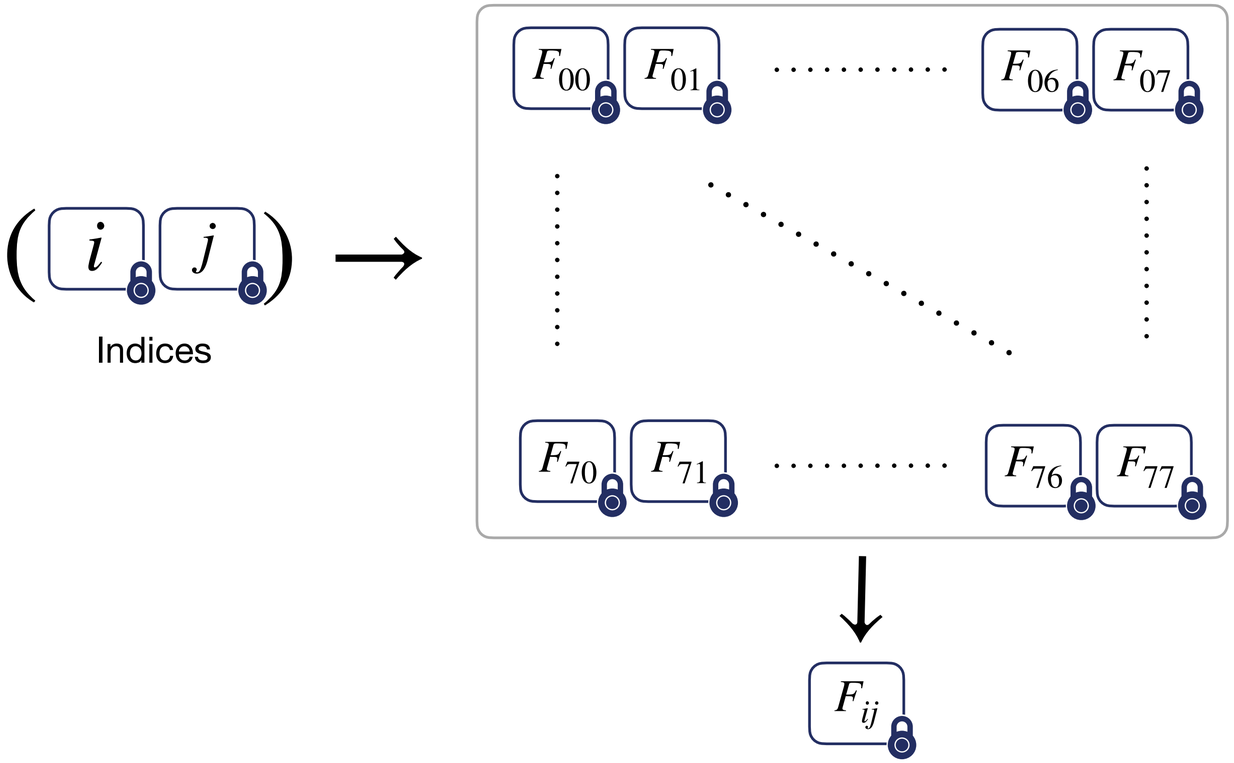


RevoLUT

A library for data-oblivious operations

ZAMA
TFHE-rs

- Blind Array Access
- Blind Matrix Access

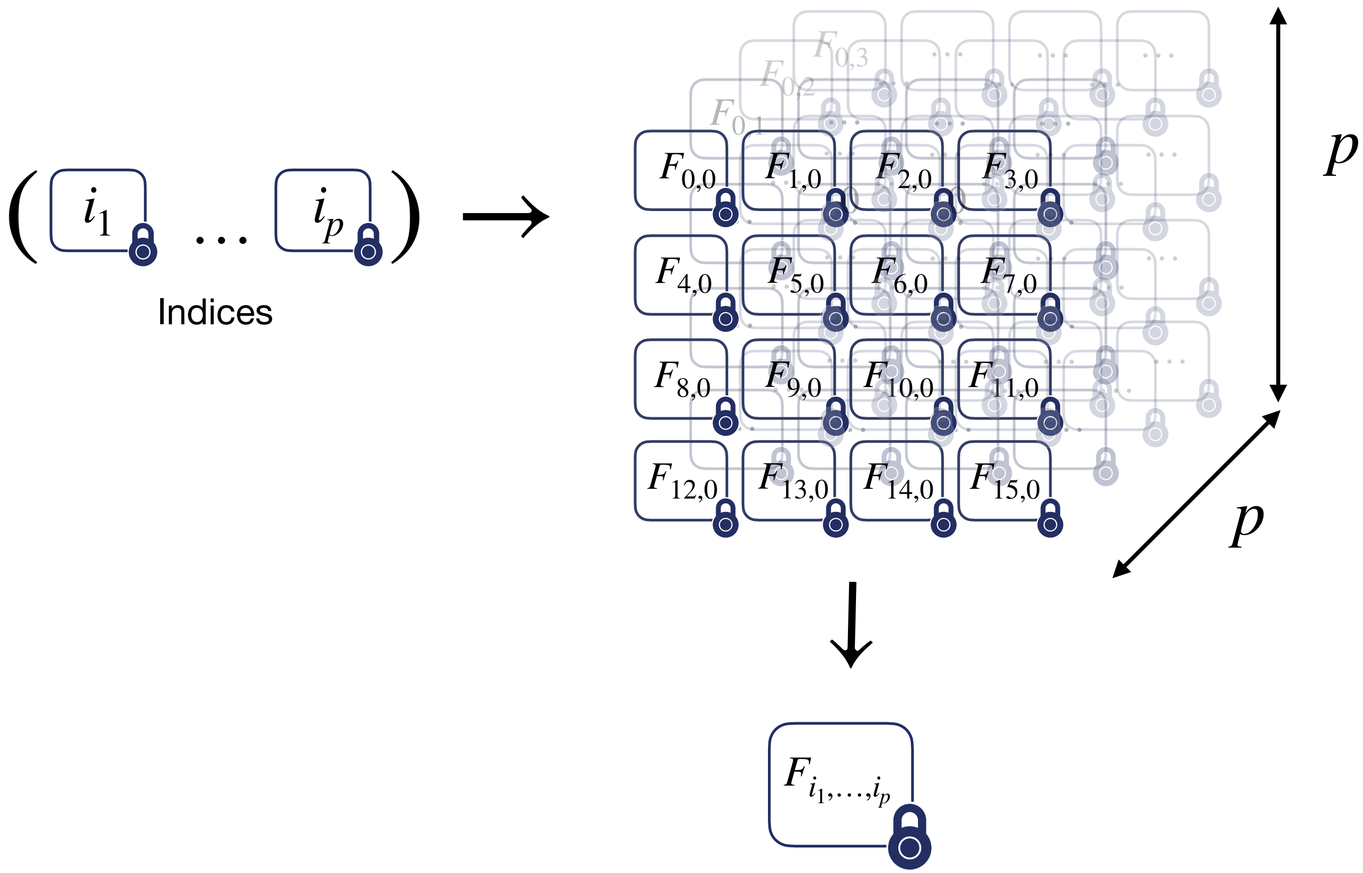


RevoLUT

A library for data-oblivious operations

ZAMA
TFHE-rs

- Blind Array Access
- Blind Matrix Access
- Blind Tensor Access

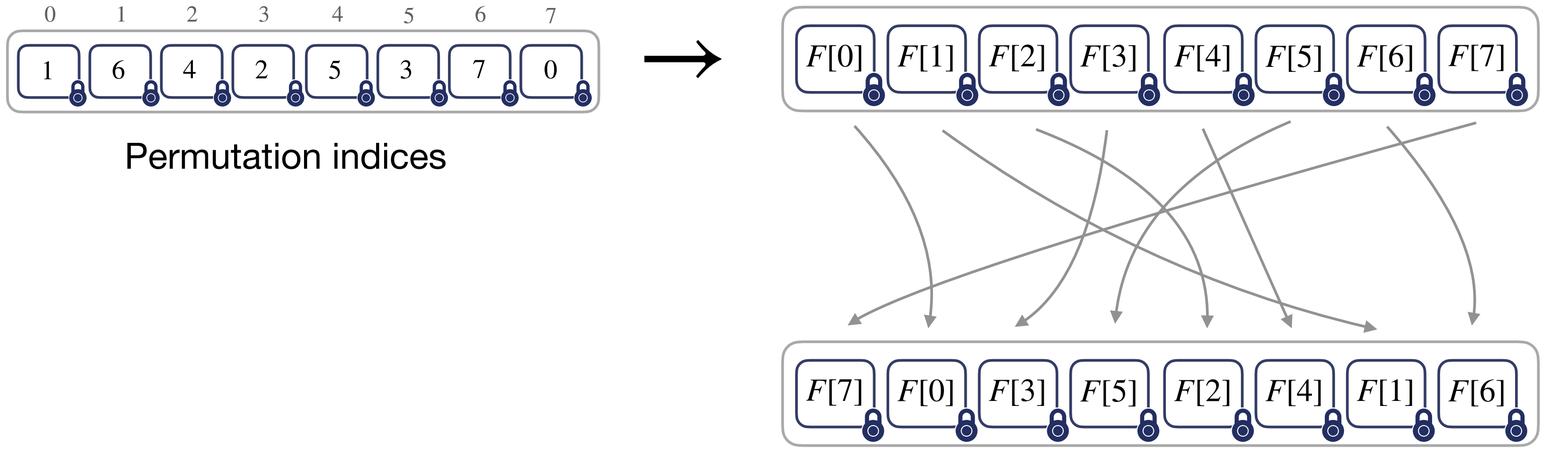


RevoLUT

A library for data-oblivious operations

ZAMA
TFHE-rs

- Blind Array Access
- Blind Matrix Access
- Blind Tensor Access
- Blind Permutation

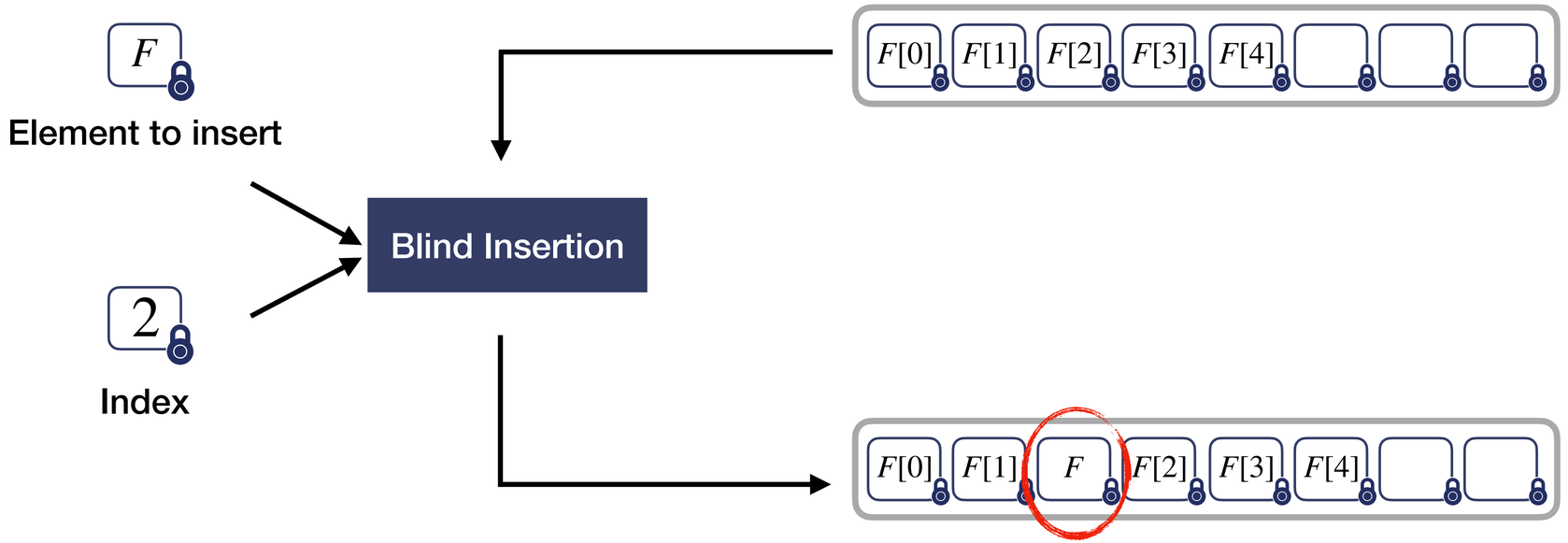


RevoLUT

A library for data-oblivious operations

ZAMA
TFHE-rs

- Blind Array Access
- Blind Matrix Access
- Blind Tensor Access
- Blind Permutation
- Blind Insertion

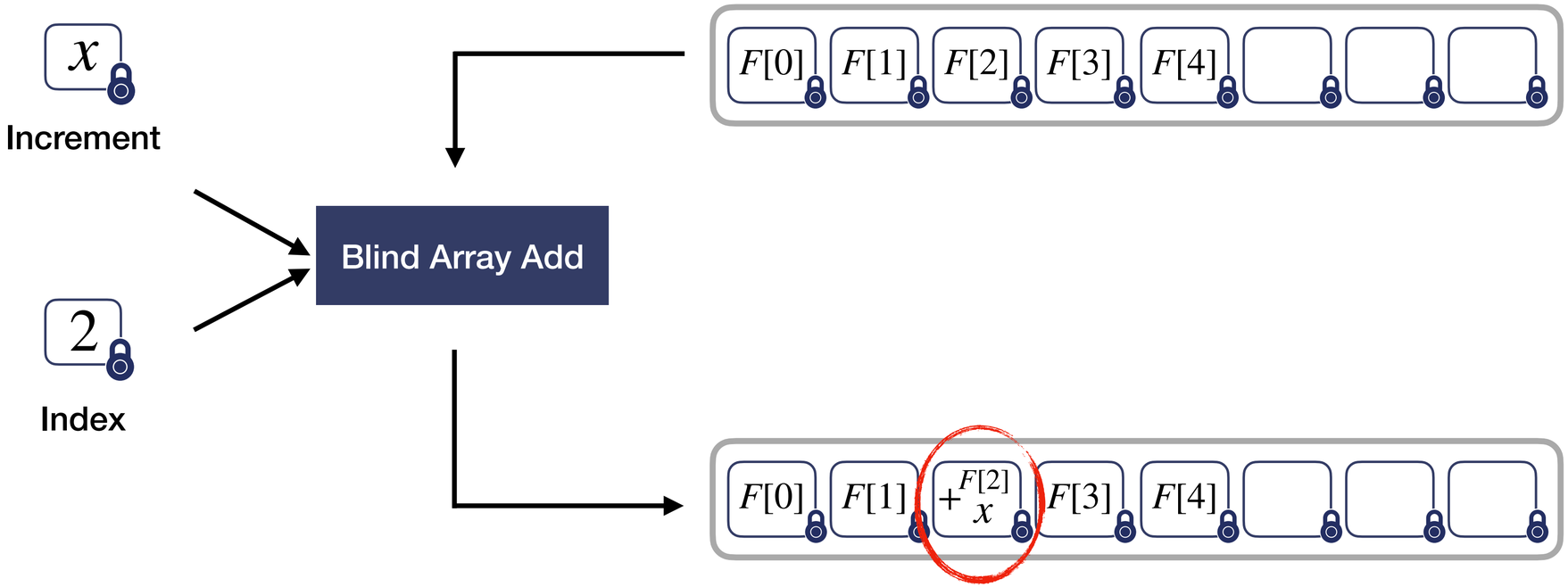


RevoLUT

A library for data-oblivious operations

ZAMA
TFHE-rs

- Blind Array Access
- Blind Matrix Access
- Blind Tensor Access
- Blind Permutation
- Blind Insertion
- Blind Array Add



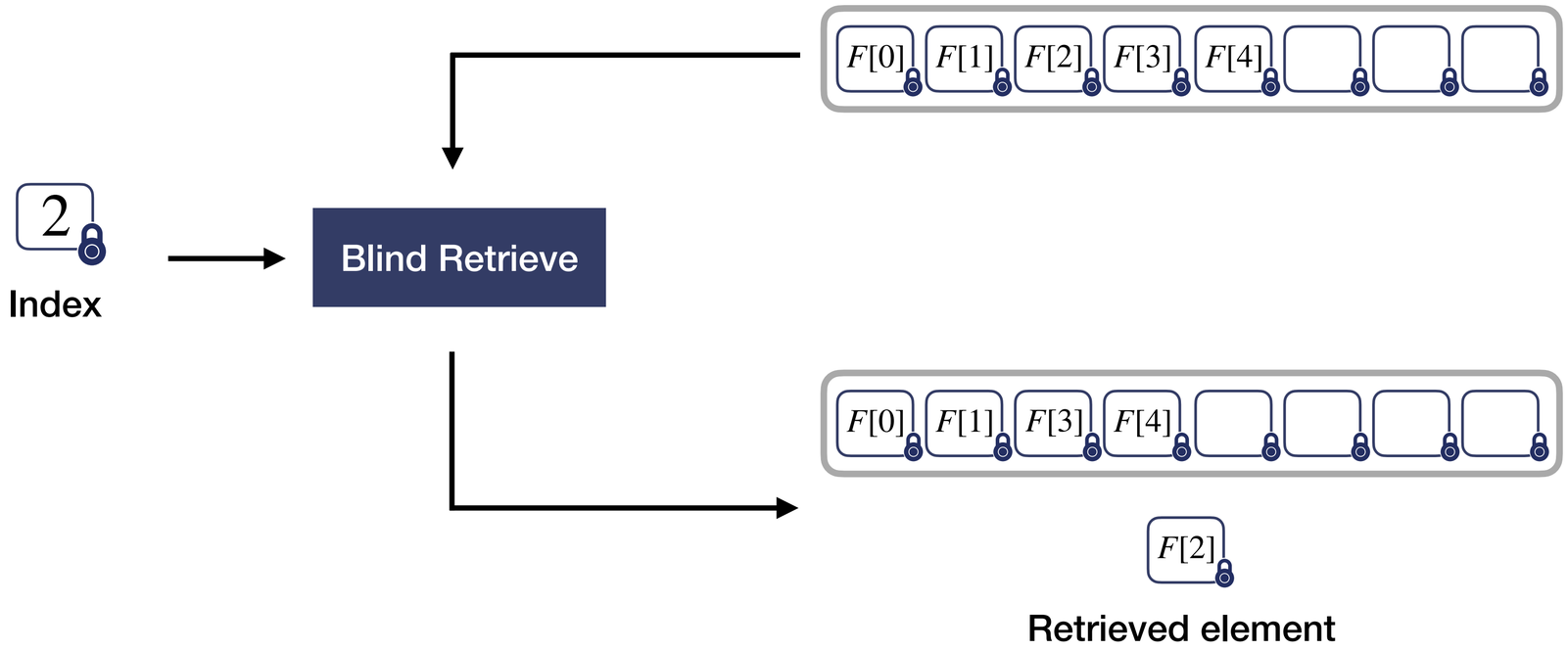
RevoLUT

A library for data-oblivious operations

ZAMA
TFHE-rs

- Blind Array Access
- Blind Matrix Access
- Blind Tensor Access
- Blind Permutation
- Blind Insertion
- Blind Array Add
- Blind Retrieve

• • •



Application to Machine Learning

Application to Machine Learning

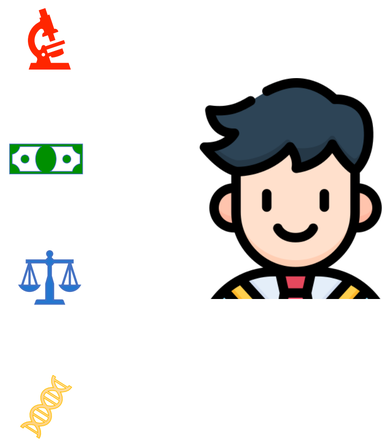
Private inference on decision tree

Application to Machine Learning

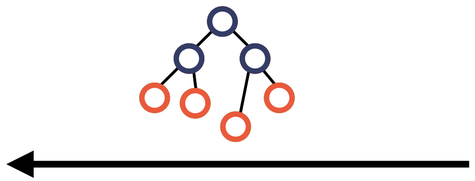
Private inference on decision tree

The naive way

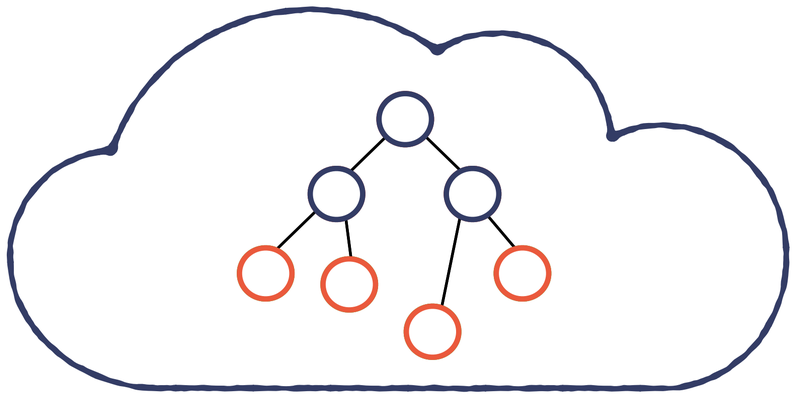
Client



Sensitive data



Cloud



- ✅ Preserves privacy of the client's data
- ❌ Preserves confidentiality of the server's model
- ✅ Needs one round of communication

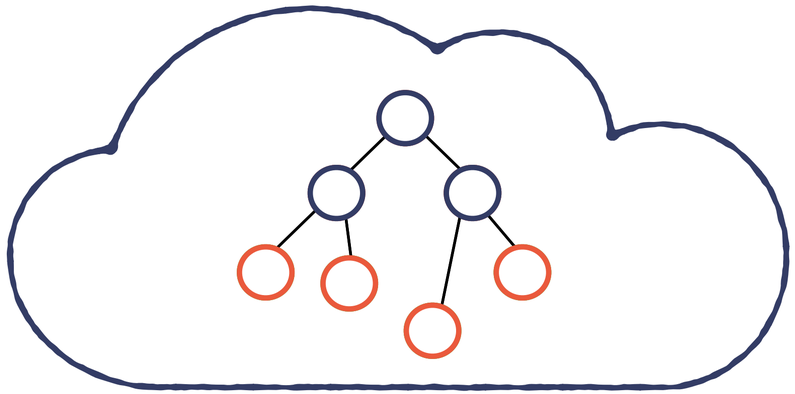
Application to Machine Learning

Private inference on decision tree

The SMPC way

Client

Cloud

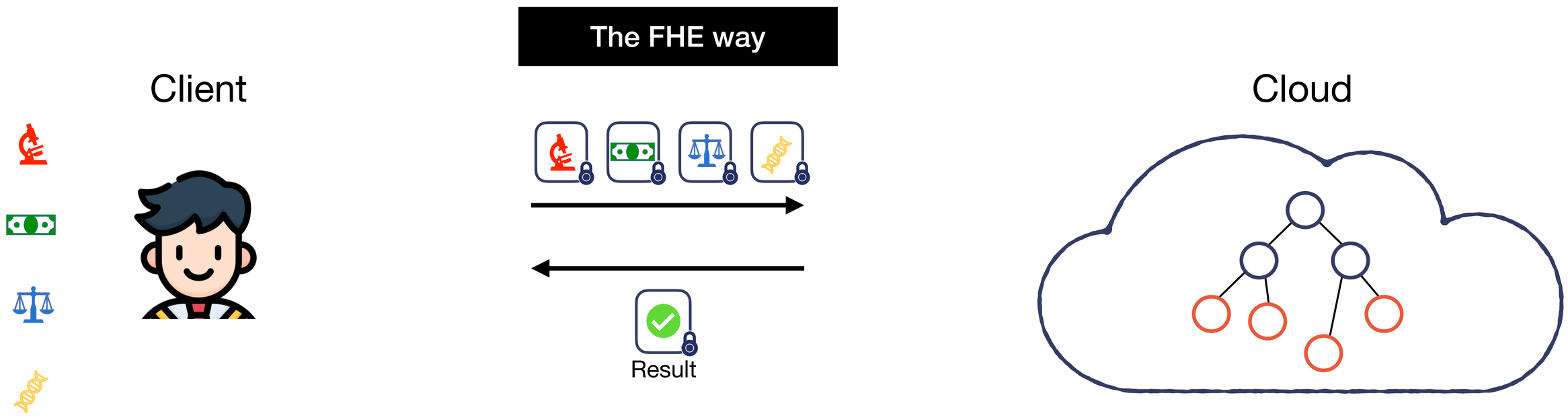


Sensitive data

- ✅ Preserves privacy of the client's data
- ✅ Preserves confidentiality of the server's model
- ❌ Needs one round of communication

Application to Machine Learning

Private inference on decision tree



- ✅ Preserves privacy of the client's data
- ✅ Preserves confidentiality of the server's model
- ✅ Needs one round of communication

Our proposal : PROBONITE

PROBONITE : PRivate One-Branch-Only Non-Interactive decision Tree Evaluation

Sofiane Azogagh
Université du Québec à Montréal (UQAM)
Montréal, Canada
azogagh.sofiane@courrier.uqam.ca

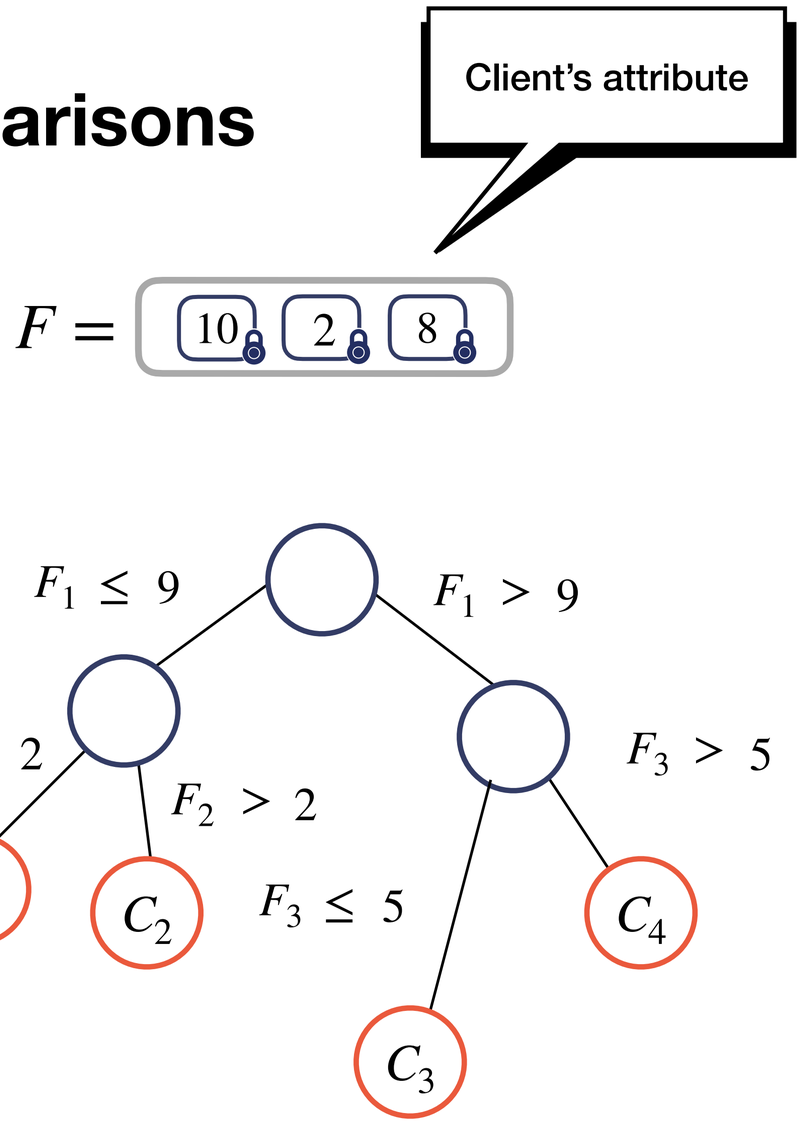
Victor Delfour
Université du Québec à Montréal (UQAM)
Montréal, Canada
delfour.victor@courrier.uqam.ca

Sébastien Gambs
Université du Québec à Montréal (UQAM)
Montréal, Canada
gambs.sebastien@uqam.ca

Marc-Olivier Killijian
Université du Québec à Montréal (UQAM)
Montréal, Canada
killijian.marc-olivier2@uqam.ca

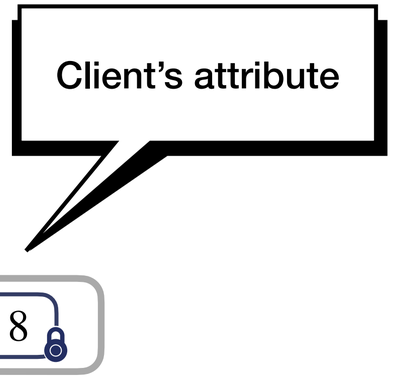
Our proposal : PROBONITE

Challenge : reducing the number of comparisons



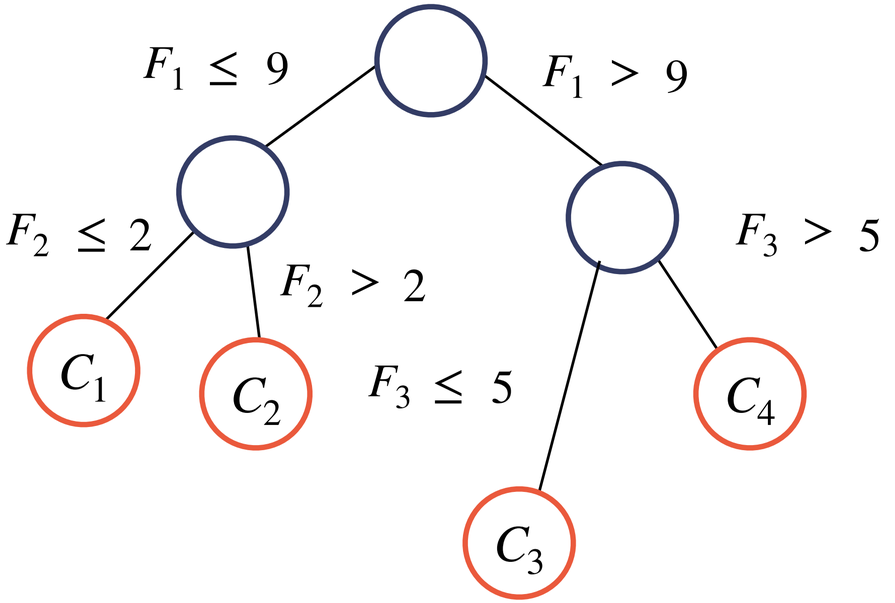
Our proposal : PROBONITE

Challenge : reducing the number of comparisons



To address this challenge, the cloud has to accomplish two tasks :

$F =$ [10] [2] [8]



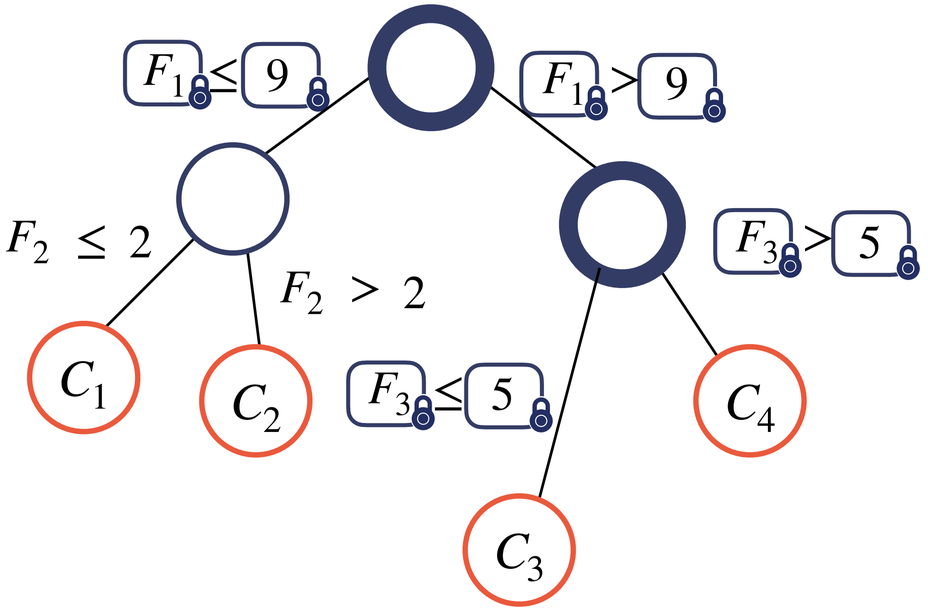
Our proposal : PROBONITE

Challenge : reducing the number of comparisons

$F =$ [10] [2] [8]

To address this challenge, the cloud has to accomplish two tasks :

1. Blindly select the node to evaluate



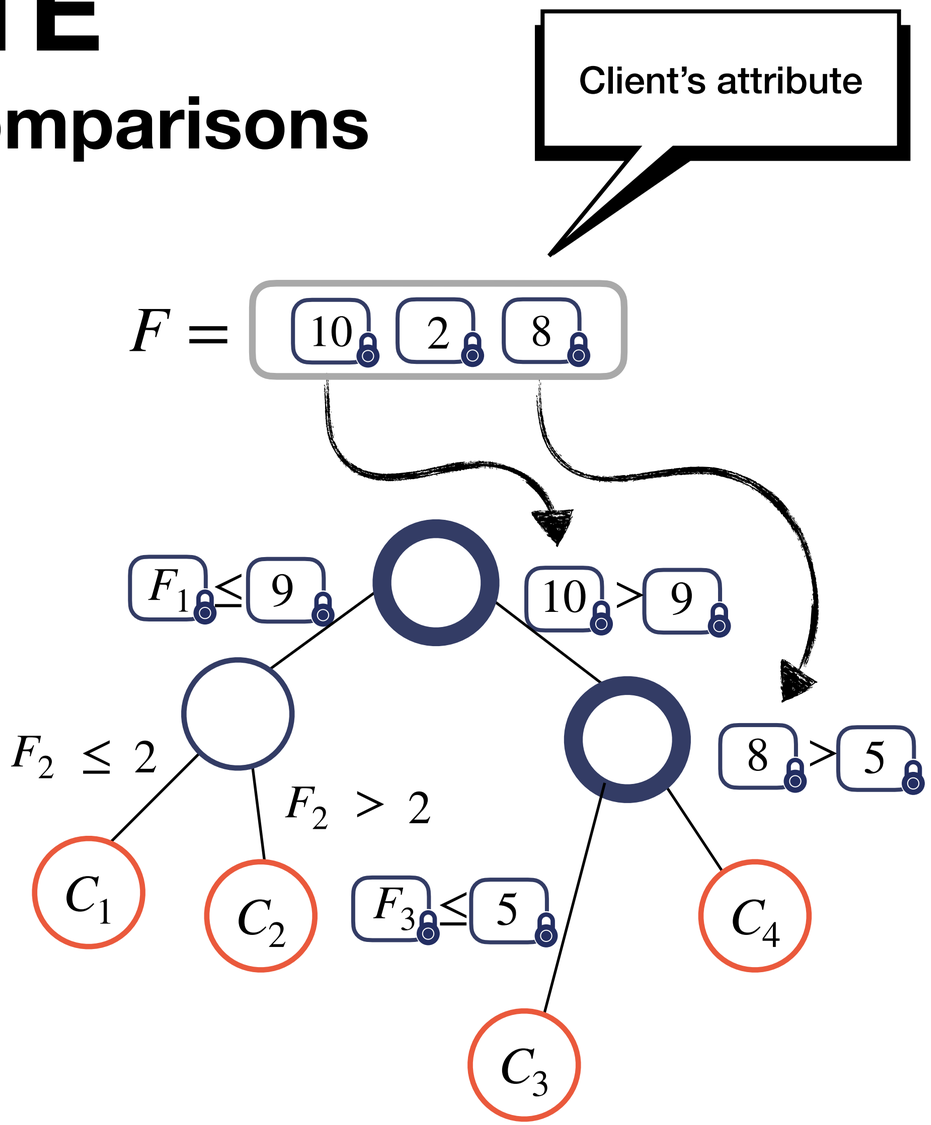
Our proposal : PROBONITE

Challenge : reducing the number of comparisons

To address this challenge, the cloud has to accomplish two tasks :

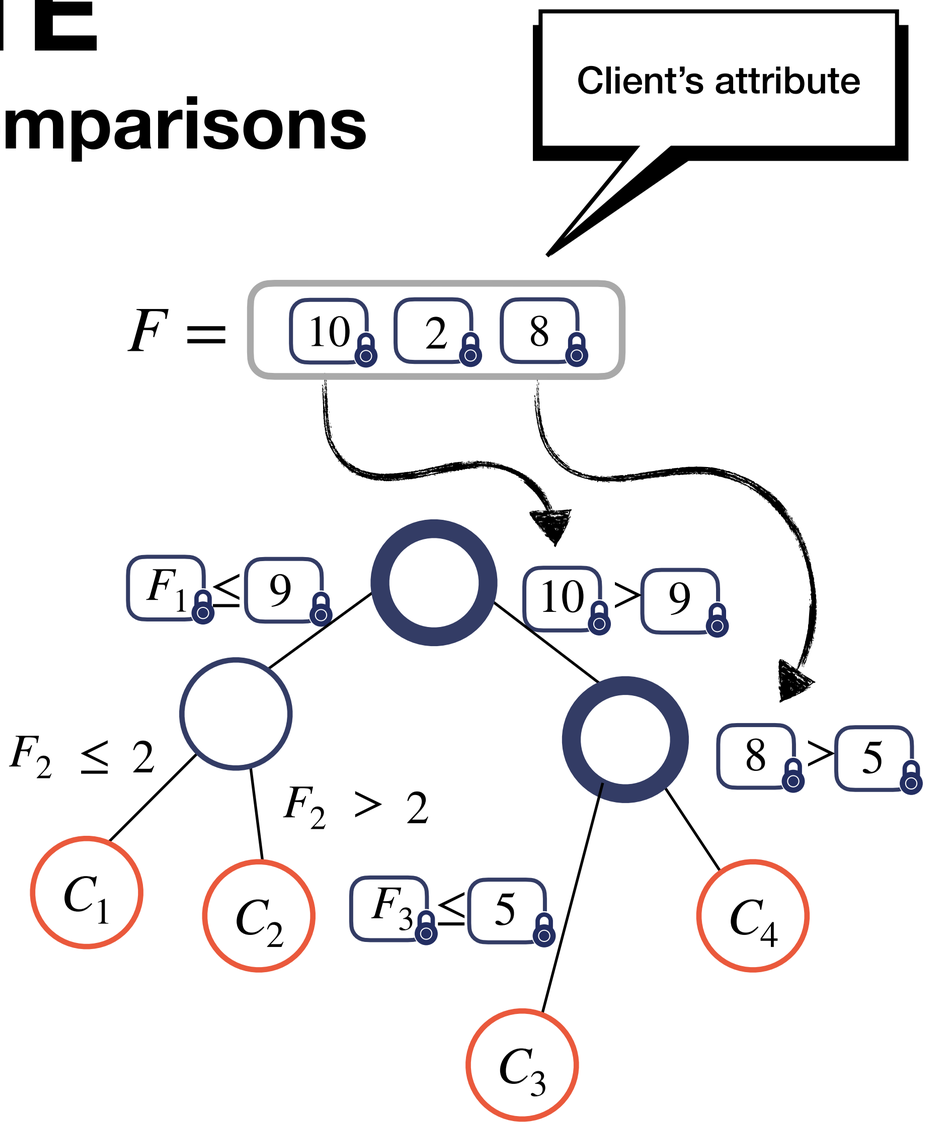
1. Blindly select the node to evaluate

2. Blindly select the attribute without getting any knowledge



Our proposal : PROBONITE

Challenge : reducing the number of comparisons



To address this challenge, the cloud has to accomplish two tasks :

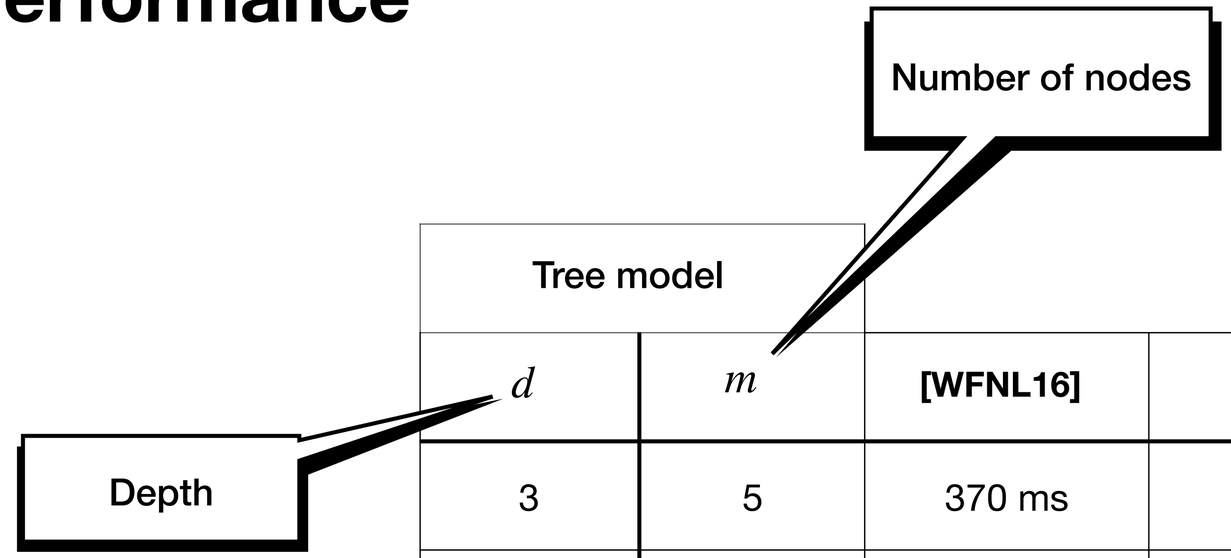
1. Blindly select the node to evaluate

2. Blindly select the attribute without getting any knowledge

Blind Array Access

Our proposal : PROBONITE

Performance



Tree model					
d	m	[WFNL16]	[LS18]	[TBK20]	PROBONITE
3	5	370 ms	590 ms	940 ms	91 ms
8	9	9671 ms	-	-	695 ms
10	500	-	53 370 ms	22 390 ms	2147 ms

[WFNL16] : David J. Wu et al. Privately Evaluating Decision Trees and Random Forests. Proc. Priv. Enhancing Technol. 2016

[LS18] : Lu W, Zhou J, Sakuma J. Non-interactive and output expressive private comparison from homomorphic encryption. AsiaCCS 2018

[TBK20] : Anselme Tueno et al. Non-interactive Private Decision Tree Evaluation. IFIP 2020

Application to Machine Learning

Private k-Nearest Neighbours

A non-comparison oblivious sort and its application to private k-NN

Sofiane Azogagh
azogagh.sofiane@courrier.uqam.ca
Univ Québec à Montréal
Canada

Marc-Olivier Killijian
killijian.marc-olivier.2@uqam.ca
Univ Québec à Montréal
Canada

Félix Larose-Gervais
larose-gervais.felix@courrier.uqam.ca
Univ Québec à Montréal
Canada

Application to Machine Learning

Private training and unlearning

Oblivious (Un)Learning of Extremely Randomized Trees*

Sofiane Azogagh
azogagh.sofiane@courrier.uqam.ca
Univ Québec à Montréal
Montréal, Canada

Zelma Aubin Birba
birba.zelma_aubin@courrier.uqam.ca
Univ Québec à Montréal
Montréal, Canada

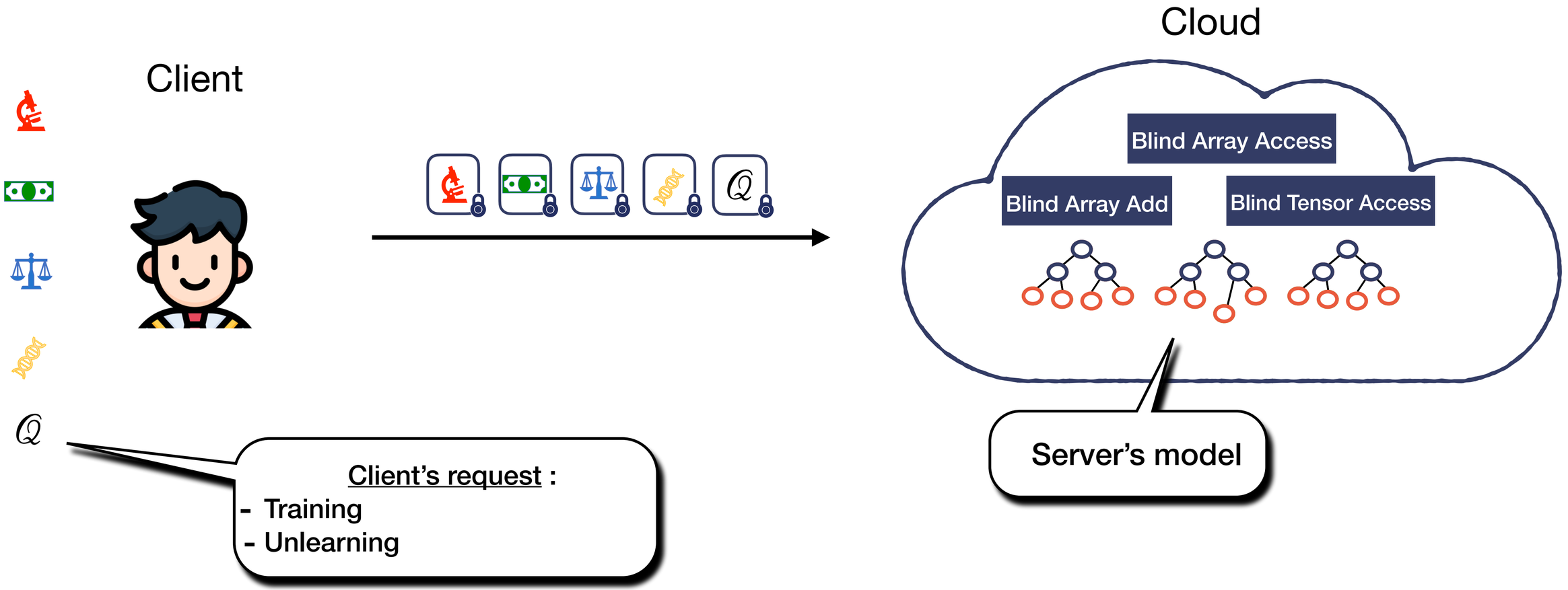
Sébastien Gambs
gambs.sebastien@courrier.uqam.ca
Univ Québec à Montréal
Montréal, Canada

Marc-Olivier Killijian
killijian.marc-olivier.2@uqam.ca
Univ Québec à Montréal
Montréal, Canada

* Under review

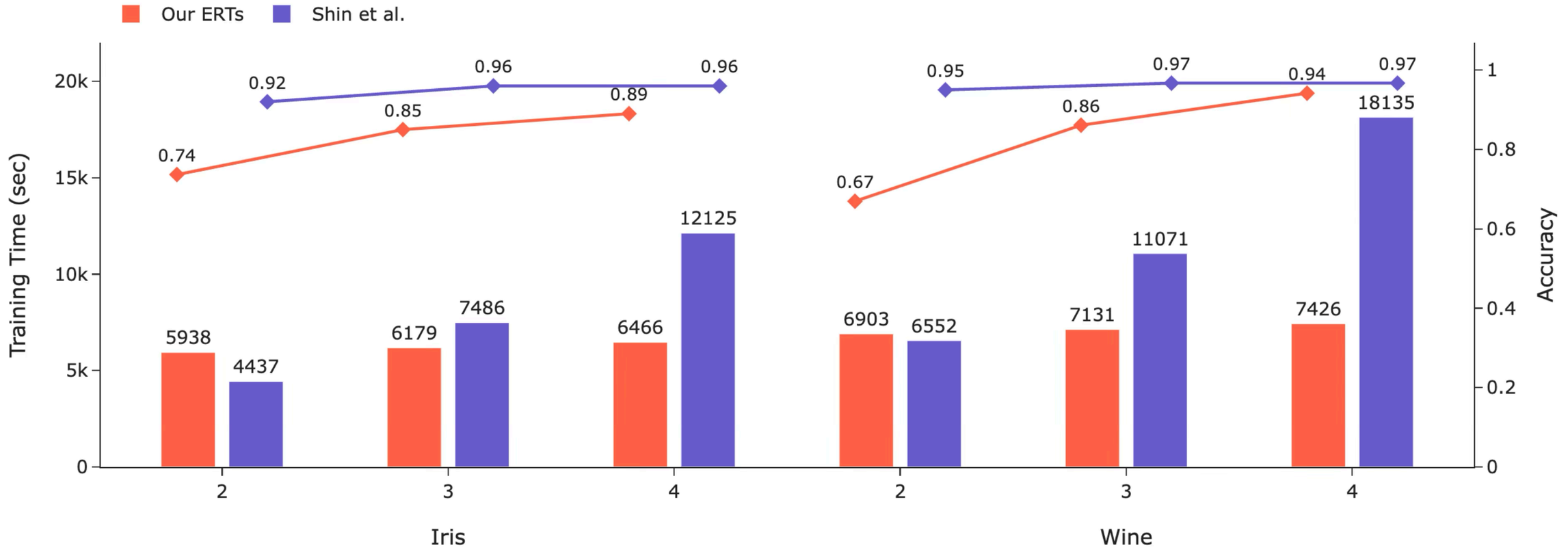
Application to Machine Learning

Private training and unlearning



Application to Machine Learning

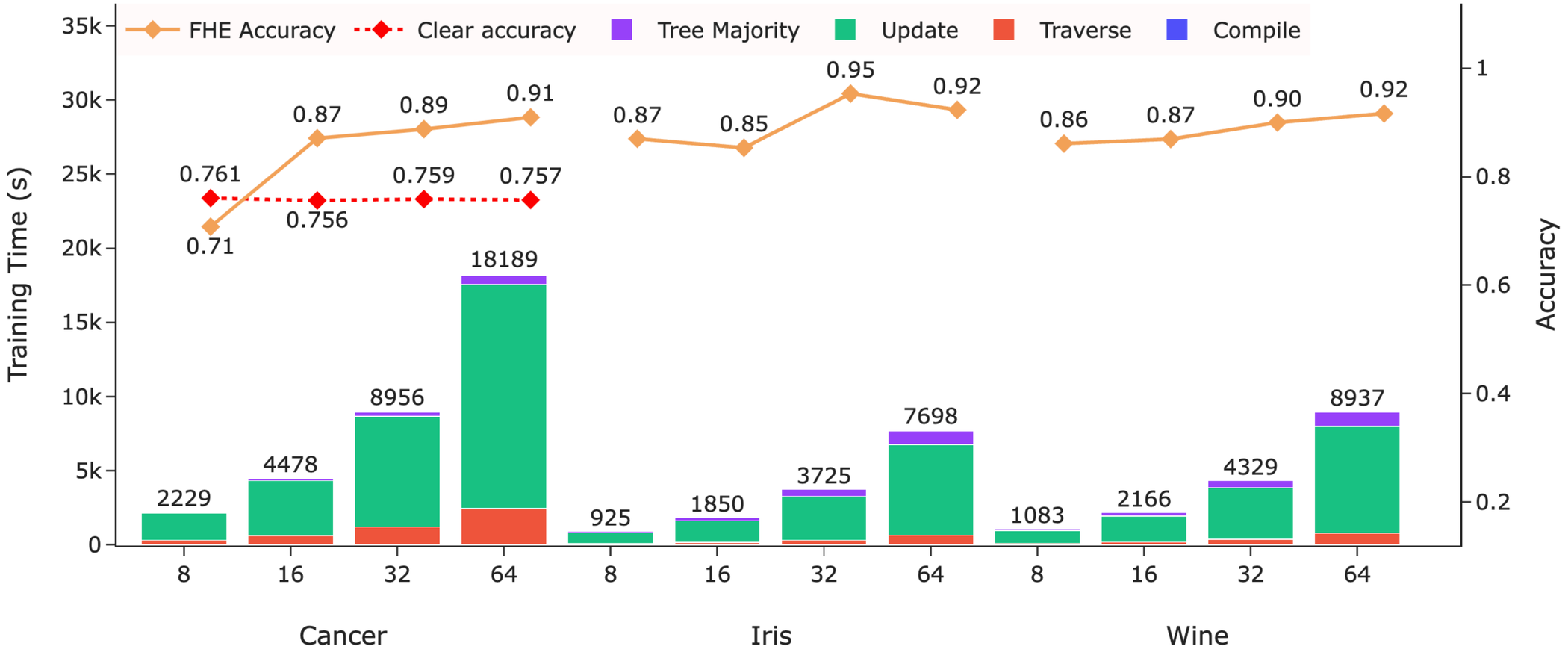
Private training and unlearning



[Shin et al.] : Fully Homomorphic Training and Inference on Binary Decision Tree and Random Forest. *Proc. ESORICS 2024*, pp. 217–237.

Application to Machine Learning

Private training and unlearning



Oblivious Turing Machine

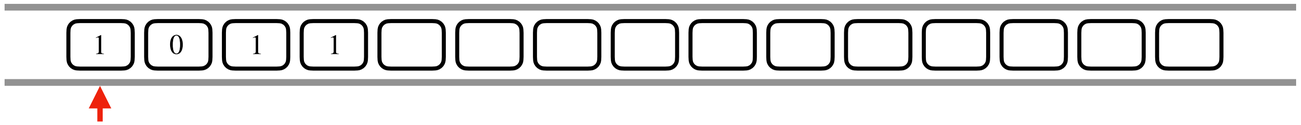
Classical Turing Machine

Example of a (binary) multiplication by 2

Instructions

State	Read	Write	Move	New State
0	0	0	R	0
	1	1	R	0
	Ø	0	N	1
1	0	0	N	1
	1	1	N	1
	Ø	Ø	N	1

Step-by-step tape



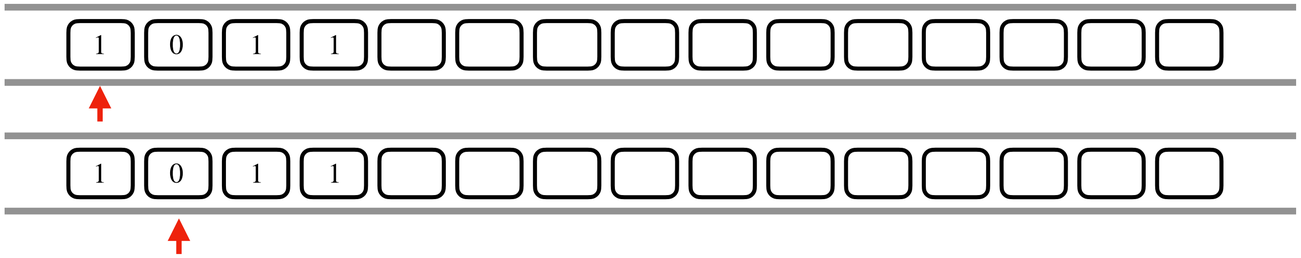
Classical Turing Machine

Example of a (binary) multiplication by 2

Instructions

State	Read	Write	Move	New State
0	0	0	R	0
	1	1	R	0
	Ø	0	N	1
1	0	0	N	1
	1	1	N	1
	Ø	Ø	N	1

Step-by-step tape



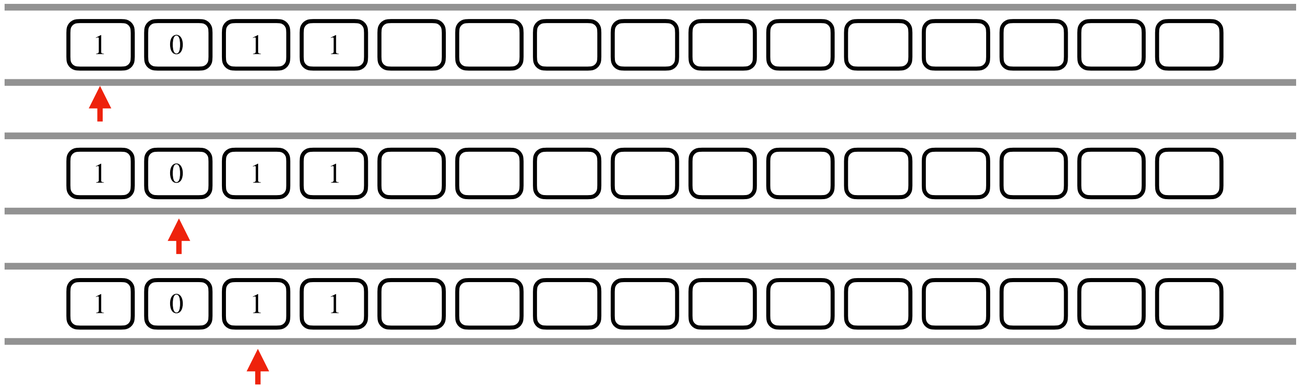
Classical Turing Machine

Example of a (binary) multiplication by 2

Instructions

State	Read	Write	Move	New State
0	0	0	R	0
	1	1	R	0
	Ø	0	N	1
1	0	0	N	1
	1	1	N	1
	Ø	Ø	N	1

Step-by-step tape



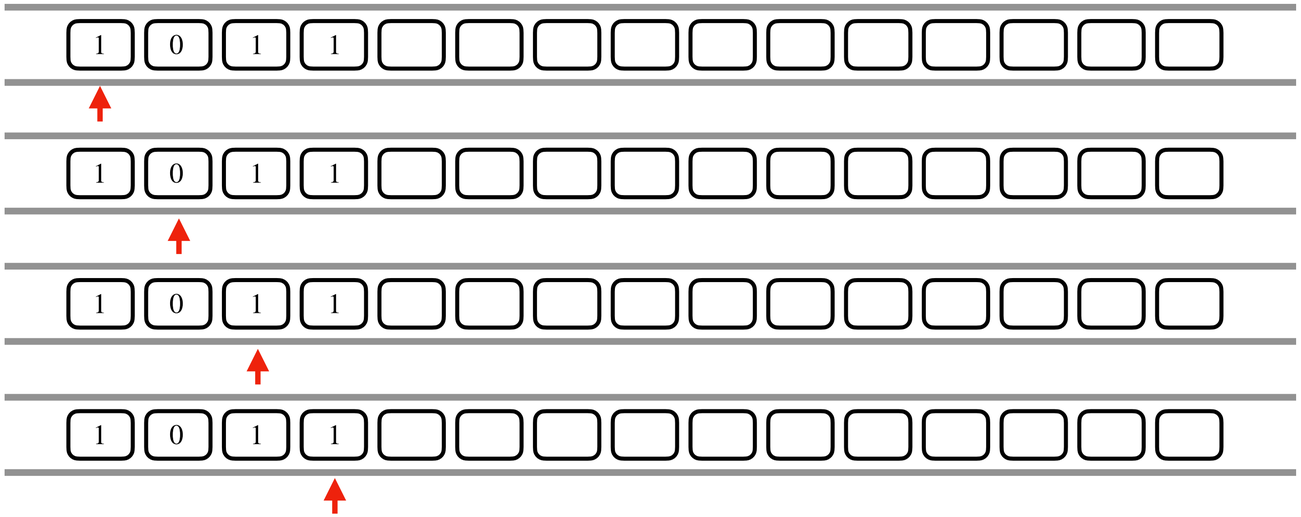
Classical Turing Machine

Example of a (binary) multiplication by 2

Instructions

State	Read	Write	Move	New State
0	0	0	R	0
	1	1	R	0
	Ø	0	N	1
1	0	0	N	1
	1	1	N	1
	Ø	Ø	N	1

Step-by-step tape



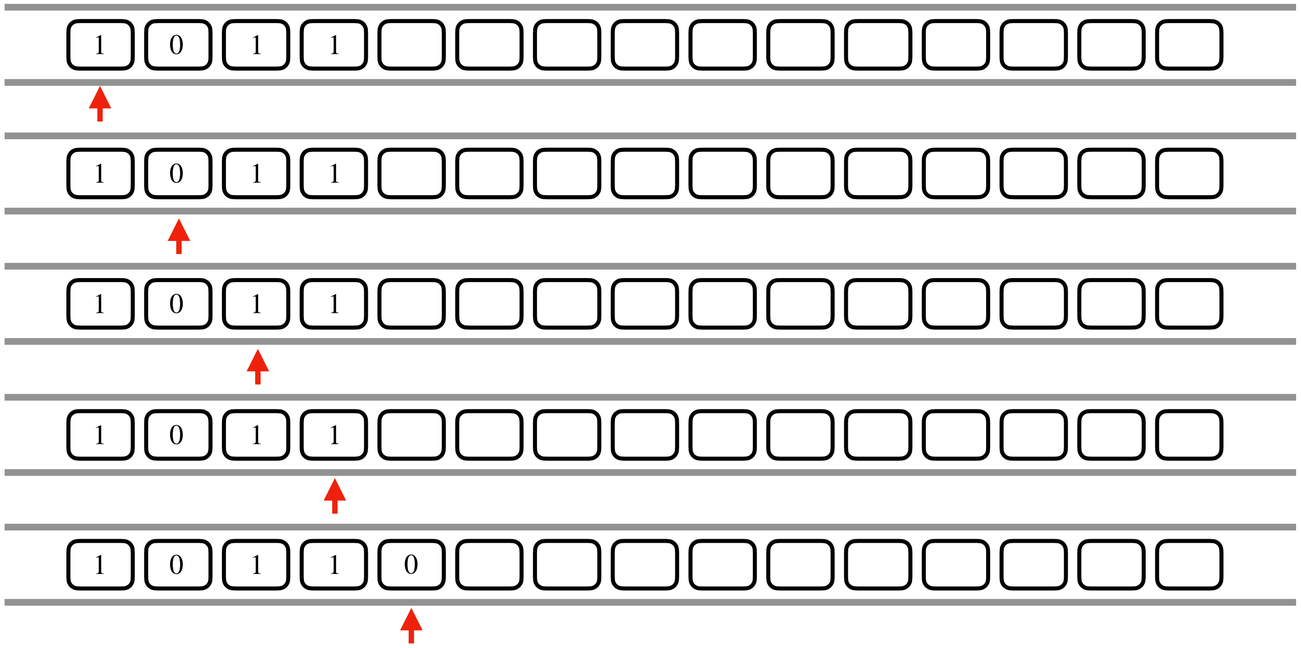
Classical Turing Machine

Example of a (binary) multiplication by 2

Instructions

State	Read	Write	Move	New State
0	0	0	R	0
	1	1	R	0
	Ø	0	N	1
1	0	0	N	1
	1	1	N	1
	Ø	Ø	N	1

Step-by-step tape



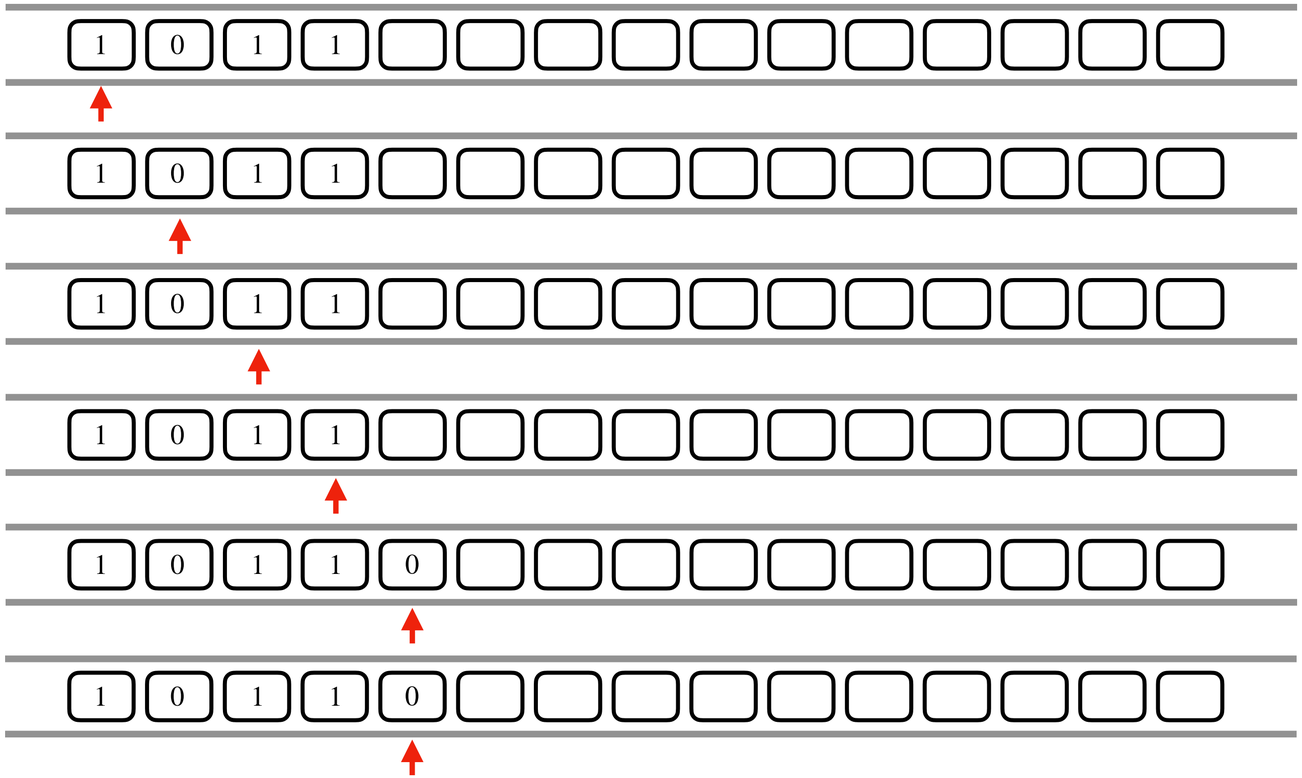
Classical Turing Machine

Example of a (binary) multiplication by 2

Instructions

State	Read	Write	Move	New State
0	0	0	R	0
	1	1	R	0
	Ø	0	N	1
1	0	0	N	1
	1	1	N	1
	Ø	Ø	N	1

Step-by-step tape



Classical Turing Machine

Example of a (binary) multiplication by 2

Instructions

State	Read	Write	Move	New State
0	0	0	R	0
	1	1	R	0
	Ø	0	N	1
1	0	0	N	1
	1	1	N	1
	Ø	Ø	N	1

0	1	0
0	1	2

I_w

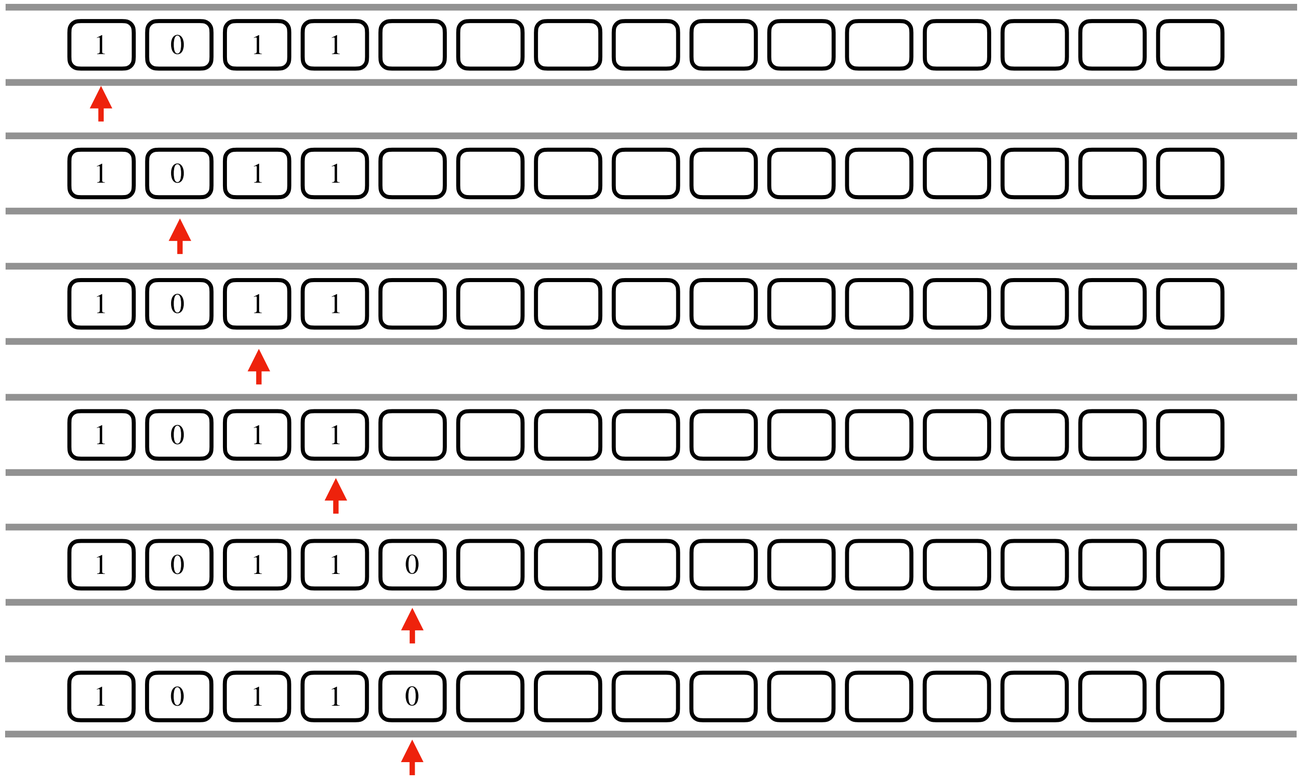
R	R	N
N	N	N

I_m

0	0	1
1	1	1

I_s

Step-by-step tape



Outsourcing the Turing Machine

Client

Server

Outsourcing the Turing Machine

Client

$f(\,.\,)$

0	1	0
0	1	2

I_w

R	R	N
N	N	N

I_m

0	0	1
1	1	1

I_s

Server

Outsourcing the Turing Machine

Client

$f(.)$

x

0	1	0
0	1	2

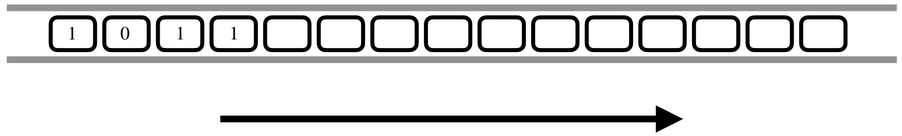
I_w

R	R	N
N	N	N

I_m

0	0	1
1	1	1

I_s



Server

Outsourcing the Turing Machine

Client

$f(\,.\,)$

x

0	1	0
0	1	2

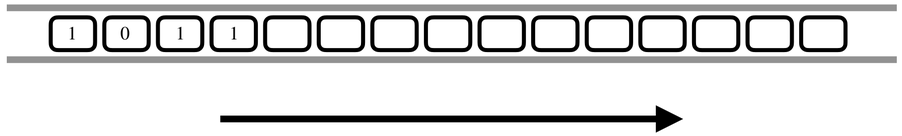
I_w

R	R	N
N	N	N

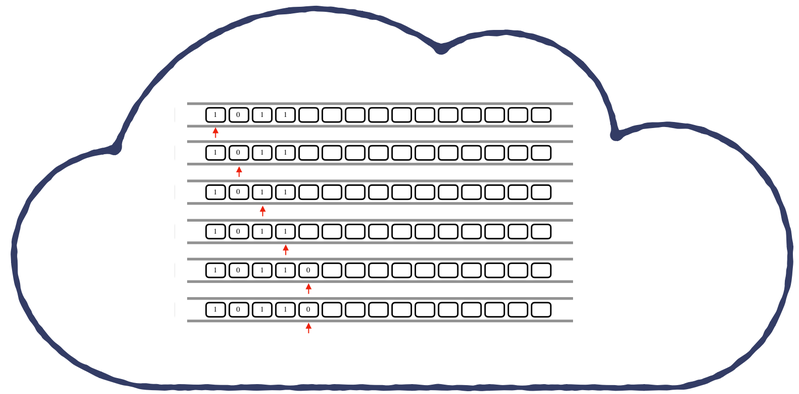
I_m

0	0	1
1	1	1

I_s



Server



Outsourcing the Turing Machine

Client

$f(\,.\,)$

x

0	1	0
0	1	2

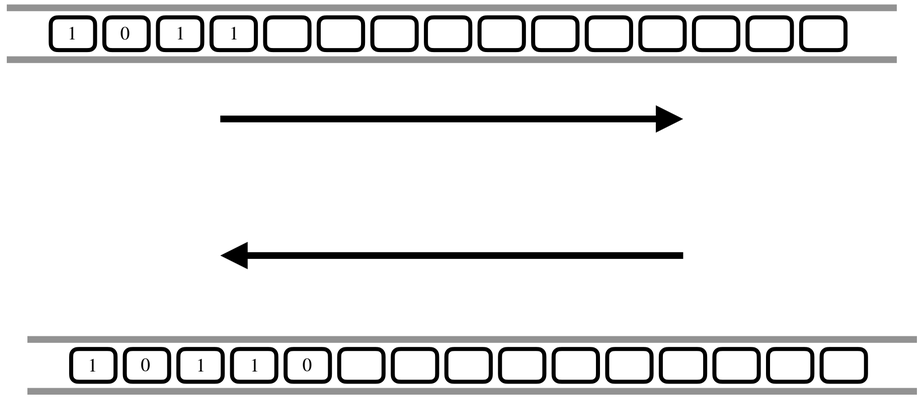
I_w

R	R	N
N	N	N

I_m

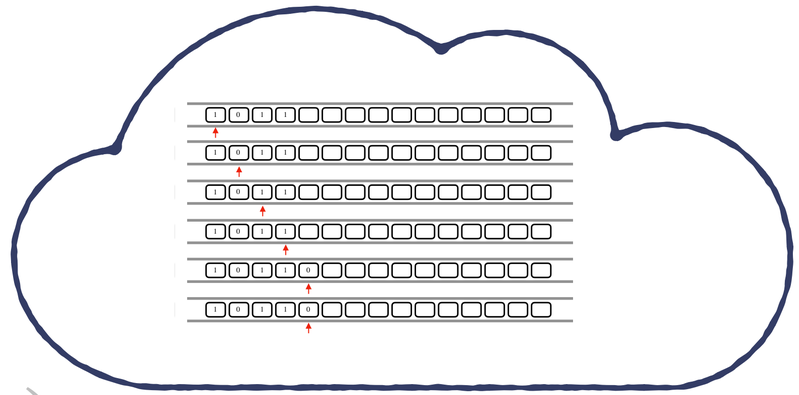
0	0	1
1	1	1

I_s



$f(x)$

Server



Our proposal : OTM

Oblivious Turing Machine*

Sofiane Azogagh
Computer Science Department
Université du Québec à Montréal
Montréal, Québec, Canada

Victor Delfour
Computer Science Department
Université du Québec à Montréal
Montréal, Québec, Canada

Marc-Olivier Killijian
Computer Science Department
Université du Québec à Montréal
Montréal, Québec, Canada

* Distinguished paper award at EDCC24

Oblivious Turing Machine

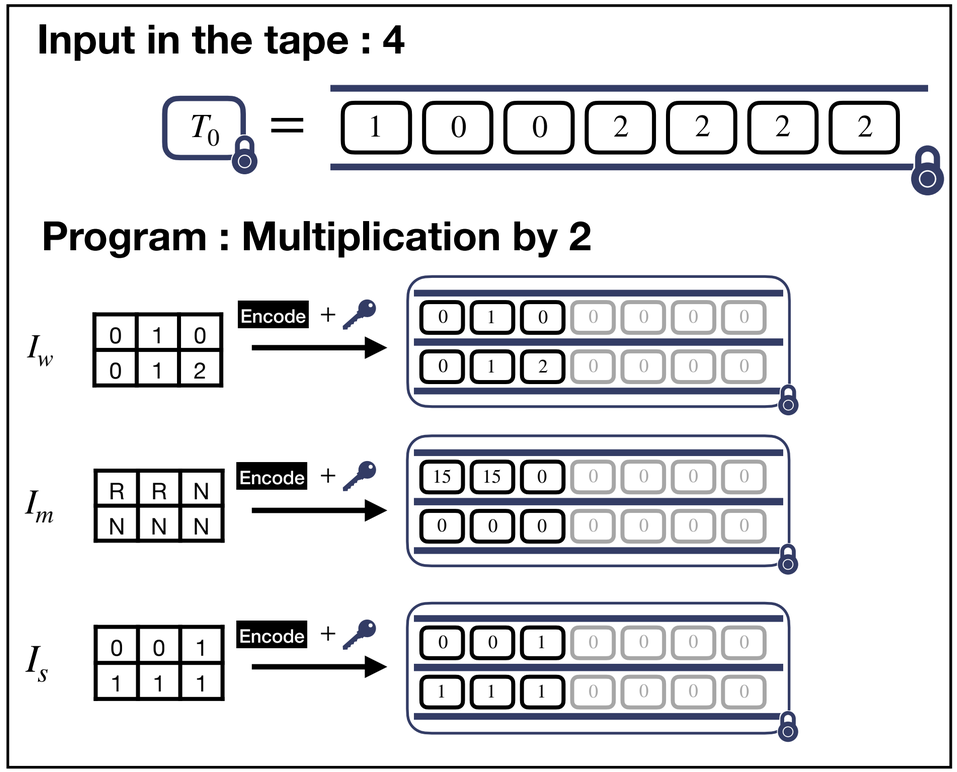
Client

Server

Oblivious Turing Machine

Client

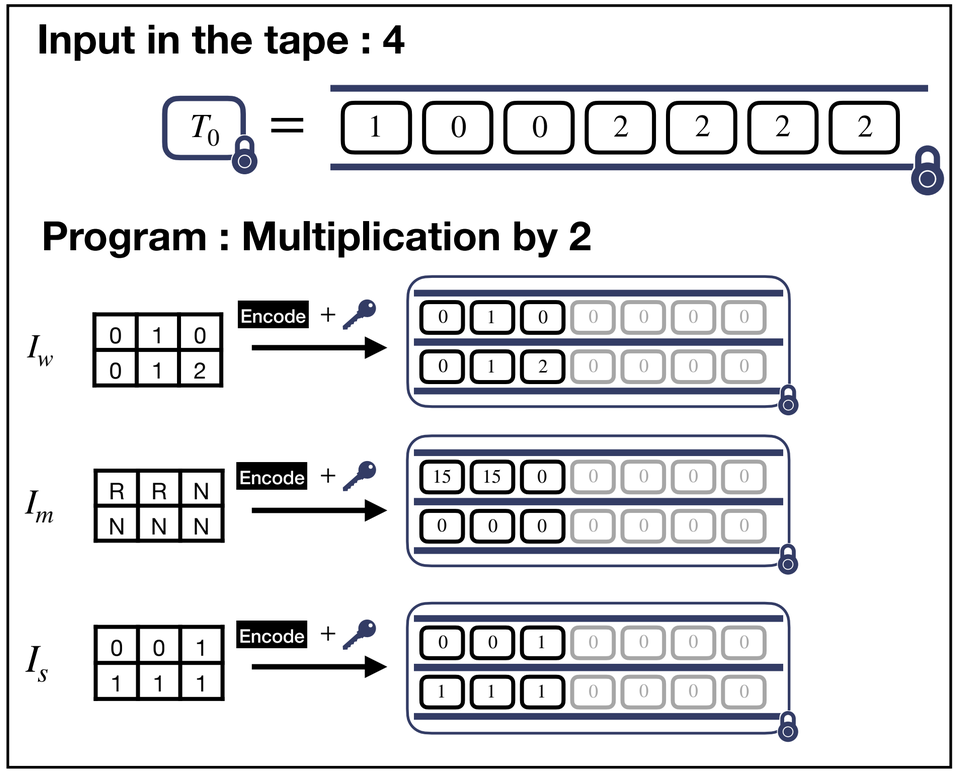
Server

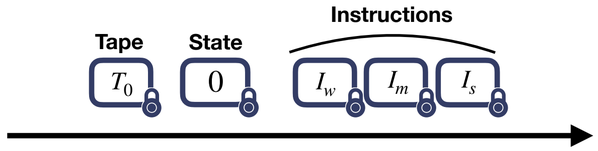


Oblivious Turing Machine

Client

Server





Oblivious Turing Machine

Client

Input in the tape : 4

T_0 = 1 0 0 2 2 2 2

Program : Multiplication by 2

I_w

0	1	0
0	1	2

Encode + 🔑 → 0 1 0 0 0 0 0 / 0 1 2 0 0 0 0

I_m

R	R	N
N	N	N

Encode + 🔑 → 15 15 0 0 0 0 0 / 0 0 0 0 0 0 0

I_s

0	0	1
1	1	1

Encode + 🔑 → 0 0 1 0 0 0 0 / 1 1 1 0 0 0 0

Tape T_0 | State 0 | Instructions I_w I_m I_s →

Server

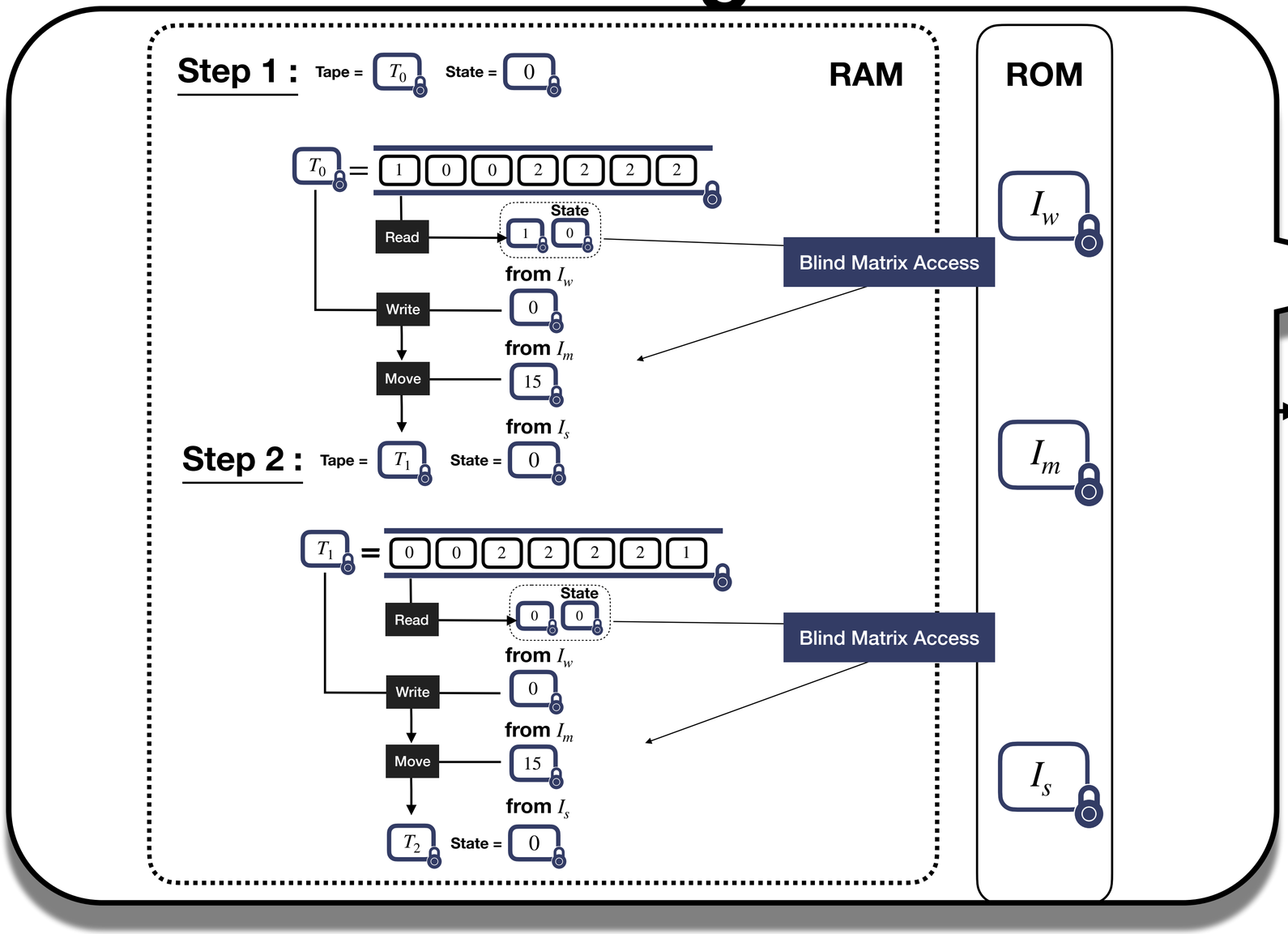
RAM

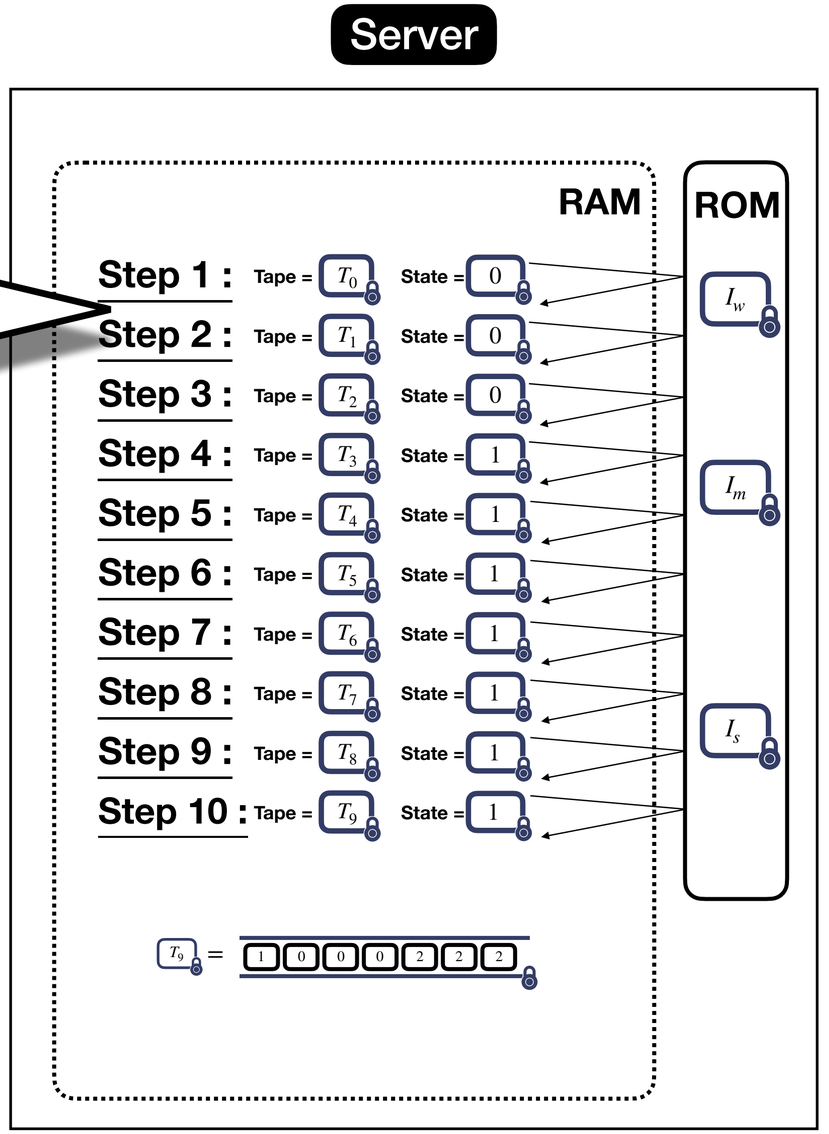
ROM I_w I_m I_s

Step 1 :	Tape = T_0	State = 0
Step 2 :	Tape = T_1	State = 0
Step 3 :	Tape = T_2	State = 0
Step 4 :	Tape = T_3	State = 1
Step 5 :	Tape = T_4	State = 1
Step 6 :	Tape = T_5	State = 1
Step 7 :	Tape = T_6	State = 1
Step 8 :	Tape = T_7	State = 1
Step 9 :	Tape = T_8	State = 1
Step 10 :	Tape = T_9	State = 1

T_9 = 1 0 0 0 2 2 2

Oblivious Turing Machine





Oblivious Turing Machine

Client

Input in the tape : 4

T_0 = 1 0 0 2 2 2 2

Program : Multiplication by 2

I_w

0	1	0
0	1	2

Encode + 🔑 → 0 1 0 0 0 0 0 / 0 1 2 0 0 0 0

I_m

R	R	N
N	N	N

Encode + 🔑 → 15 15 0 0 0 0 0 / 0 0 0 0 0 0 0

I_s

0	0	1
1	1	1

Encode + 🔑 → 0 0 1 0 0 0 0 / 1 1 1 0 0 0 0

Tape T_0 | State 0 | Instructions I_w I_m I_s →

Server

RAM **ROM**

Step 1 :	Tape = T_0	State = 0		I_w
Step 2 :	Tape = T_1	State = 0		
Step 3 :	Tape = T_2	State = 0		
Step 4 :	Tape = T_3	State = 1		I_m
Step 5 :	Tape = T_4	State = 1		
Step 6 :	Tape = T_5	State = 1		
Step 7 :	Tape = T_6	State = 1		
Step 8 :	Tape = T_7	State = 1		I_s
Step 9 :	Tape = T_8	State = 1		
Step 10 :	Tape = T_9	State = 1		

T_9 = 1 0 0 0 2 2 2

Oblivious Turing Machine

Client

Server

Input in the tape : 4

T_0 = 1 0 0 2 2 2 2

Program : Multiplication by 2

I_w | 0 1 0 / 0 1 2 | Encode + key → 0 1 0 0 0 0 0 / 0 1 2 0 0 0 0

I_m | R R N / N N N | Encode + key → 15 15 0 0 0 0 0 / 0 0 0 0 0 0 0

I_s | 0 0 1 / 1 1 1 | Encode + key → 0 0 1 0 0 0 0 / 1 1 1 0 0 0 0

Tape T_0 State 0 Instructions I_w I_m I_s

Final Tape T_9 State 1

1 0 0 0 2 2 2 → key → 1 0 0 0 2 2 2

RAM

ROM (I_w, I_m, I_s)

Step 1 : Tape = T_0 State = 0

Step 2 : Tape = T_1 State = 0

Step 3 : Tape = T_2 State = 0

Step 4 : Tape = T_3 State = 1

Step 5 : Tape = T_4 State = 1

Step 6 : Tape = T_5 State = 1

Step 7 : Tape = T_6 State = 1

Step 8 : Tape = T_7 State = 1

Step 9 : Tape = T_8 State = 1

Step 10 : Tape = T_9 State = 1

T_9 = 1 0 0 0 2 2 2

Conclusion

Conclusion

- RevoLUT is an efficient way to leverage TFHE's LUT as core data structures for secure and oblivious computation.
- It has proven its efficiency across various applications in Machine Learning
- We are exploring new applications across domains, so feel free to reach out and collaborate!

ia.cr/2024/1935

Poster at FHE.org 2025

sofianeazogagh/revoLUT

Thank you !

Any questions ?

azogagh.sofiane@courrier.uqam.ca

sofianeazogagh.github.io