Combining Cryptography and Other Techniques for Various Privacy-Preserving Applications

Mariya Georgieva Belorgey, Sergiu Carpov

Presented at the NIST Crypto Reading Club on 2024-May-15

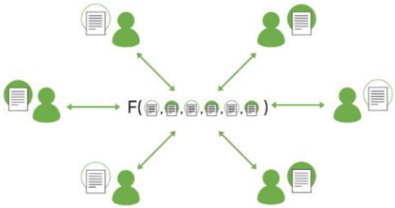
Privacy-Enhancing Technologies

Privacy-Enhancing Technologies (PETs)

inpher

Primary Goal:

They enable the computation of an arbitrary function without revealing the input data.



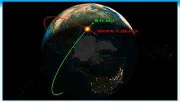
Examples of PETS

- Fully Homomorphic Encryption (FHE)
- Secure Multi-Party Computation (MPC)
- Federated Learning (FL)
- Differential Privacy (DP)
- Trusted Execution Environments (TEEs)

PETs use-cases

inpher

Physics/Astronautics



- Predict trajectories: are satellites on a collision course?
- **Iridium 33 and Kosmos-2251 Satellite Collision in 2009**
- Need to evaluate non-linear functions with high precision on secret trajectories

Medicine/Genomic

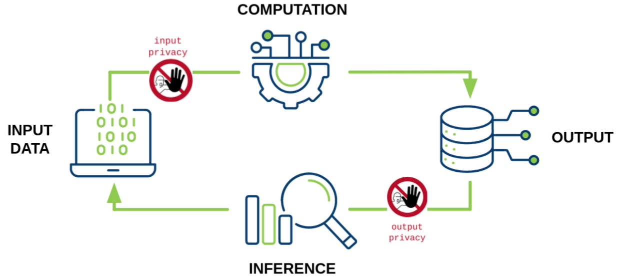
- Predictive healthcare
- Find the right dosage for a cure
- Need to evaluate/train machine learning algorithms on secret medical data

Finance/Banking

- Fraud detection, risk scoring
- Investment Banking and Hedge Funds
- Detect loops in transaction graphs

Input vs Output Privacy

inpher



Input Privacy

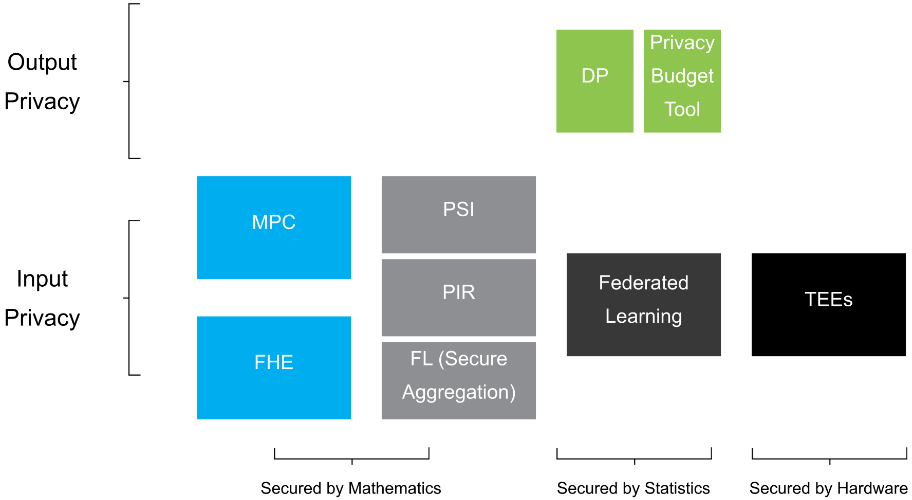
Allows forward computation from input data without disclosing it.

Output Privacy

Privacy prevents backward inference from disclosed output results.

PETs

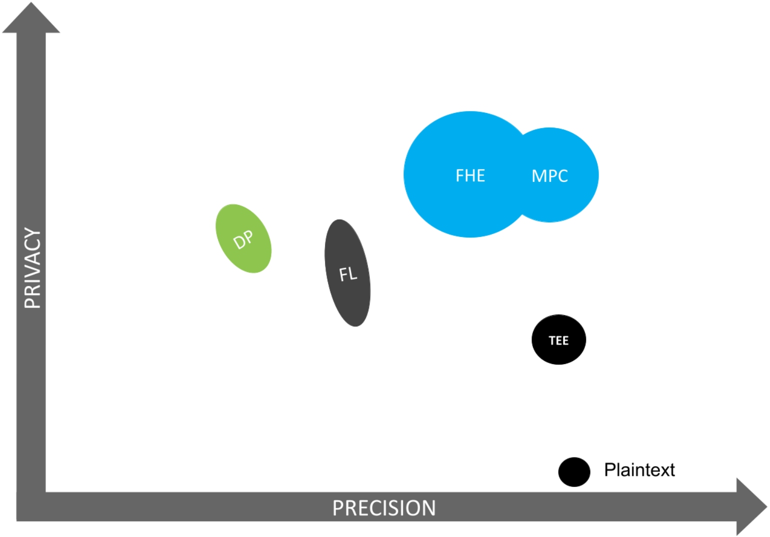
inpher



*inspired by N. Smart

PETs

inpher



Fully Homomorphic Encryption

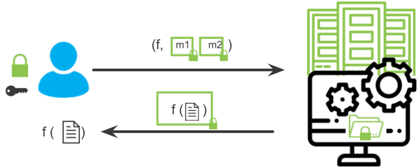
Fully Homomorphic Encryption

inpher

Given ciphertexts $(c_1, c_2, \ldots, c_k) = (\mathcal{E}(m_1), \mathcal{E}(m_2), \ldots, \mathcal{E}(m_k))$

The homomorphic computation consists of computing $\mathcal{E}(f(m_1, m_2, \ldots, m_k))$ without decryption

A scheme that can homomorphically evaluate all function is said to be **fully homomorphic**



Examples

- Multiplicatively homomorphic : RSA
  $c_1 = m_1^e \mod N$ et $c_2 = m_2^e \mod N$
  $c_1.c_2 = (m_1.m_2)^e \mod N = \mathcal{E}(m_1.m_2) \mod N$
- Additively homomorphic : Paillier
  $c_1 = g^{m_1} r_1^n \mod n^2$ et $c_2 = g^{m_2} r_2^n \mod n^2$
  $c_1.c_2 = g^{m_1+m_2}(r_1.r_2)^n \mod n^2 = \mathcal{E}(m_1 + m_2) \mod n^2$

Fully Homomorphic Encryption

inpher

Given ciphertexts $(c_1, c_2, \ldots, c_k) = (\mathcal{E}(m_1), \mathcal{E}(m_2), \ldots, \mathcal{E}(m_k))$

The homomorphic computation consists of computing $\mathcal{E}(f(m_1, m_2, \ldots, m_k))$ without decryption

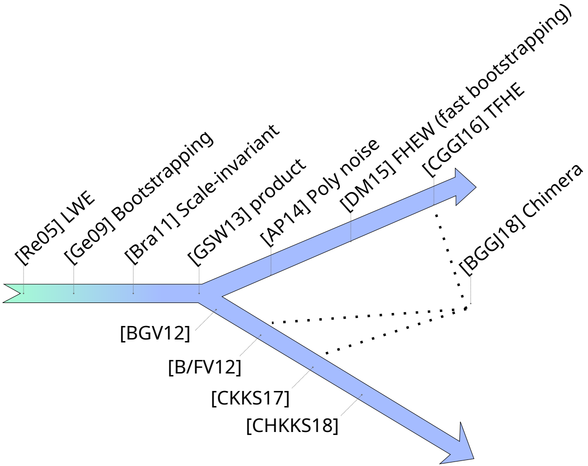
A scheme that can homomorphically evaluate all function is said to be
fully homomorphic

Examples

- Multiplicatively homomorphic : RSA
  $c_1 = m_1^e \mod N$ et $c_2 = m_2^e \mod N$
  $c_1.c_2 = (m_1.m_2)^e \mod N = \mathcal{E}(m_1.m_2) \mod N$
- Additively homomorphic : Paillier
  $c_1 = g^{m_1} r_1^n \mod n^2$ et $c_2 = g^{m_2} r_2^n \mod n^2$
  $c_1.c_2 = g^{m_1+m_2}(r_1.r_2)^n \mod n^2 = \mathcal{E}(m_1 + m_2) \mod n^2$

Little history of FHE

inpher



Almost all FHE are based on LWE/RLWE problems.

We distinguish two main families of homomorphic encryption schemes

Bootstrapped constructions

Set the parameters, it is possible to homomorphically evaluate any function.

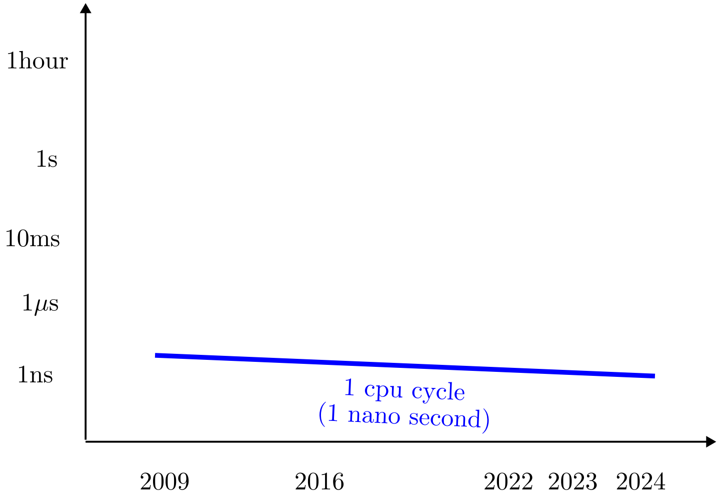
- ✔ No depth limitations
- ✔ Fast single evaluation
- ✘ Slow multiple evaluations

Leveled constructions

Set the function, there exists parameters to homomorphically evaluate it.

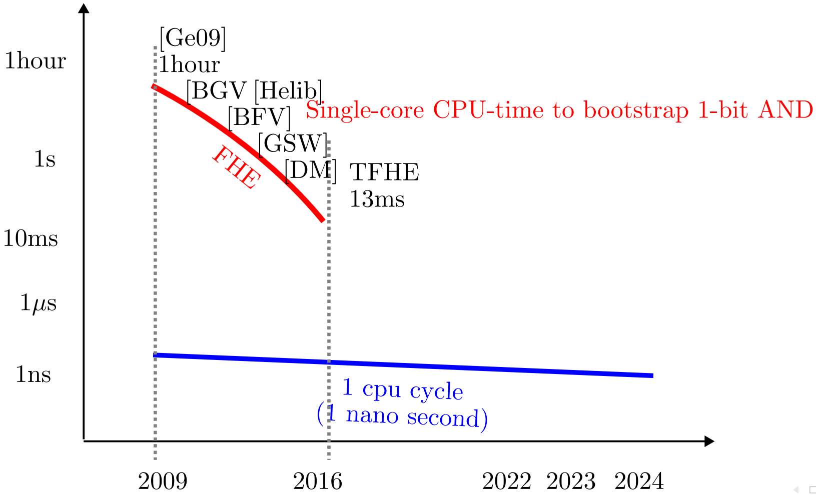
- ✘ The depth has to be known in advance
- ✘ Slow single evaluations
- ✔ Fast multiple evaluations

Bootstrapping performance evolution



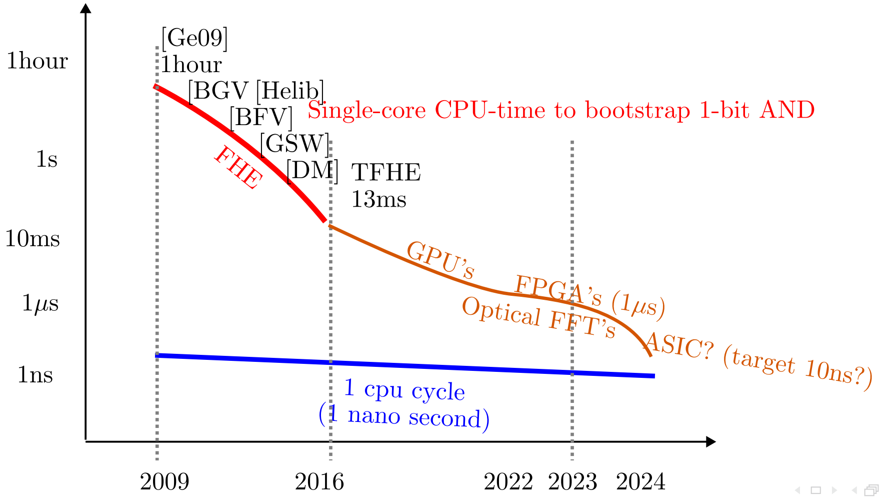
Bootstrapping performance evolution

inpher



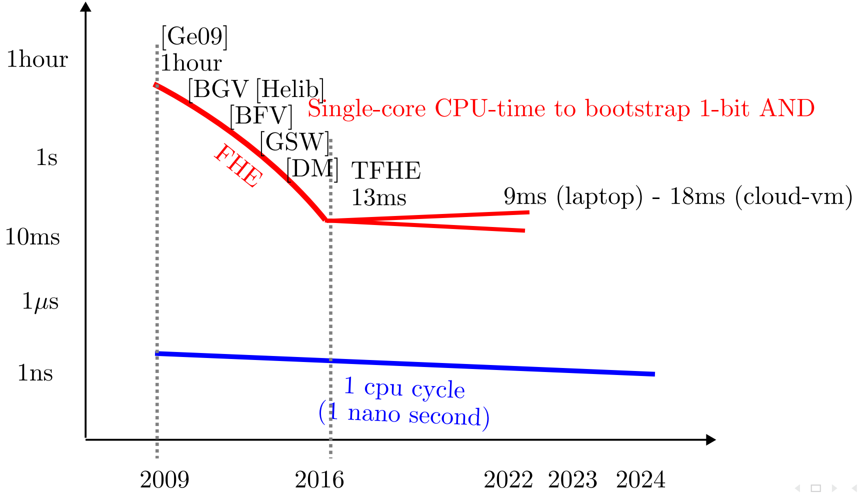
Bootstrapping performance evolution

inpher



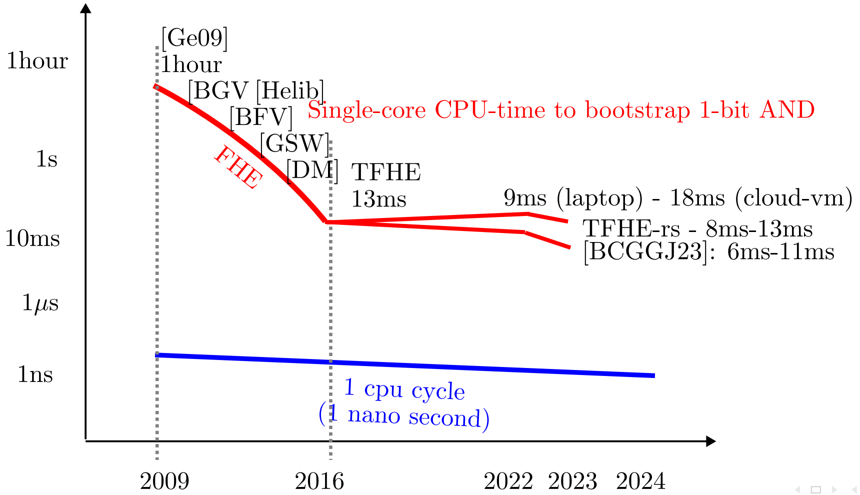
Bootstrapping performance evolution

inpher



Bootstrapping performance evolution

inpher



Fully Homomorphic Schemes

inpher

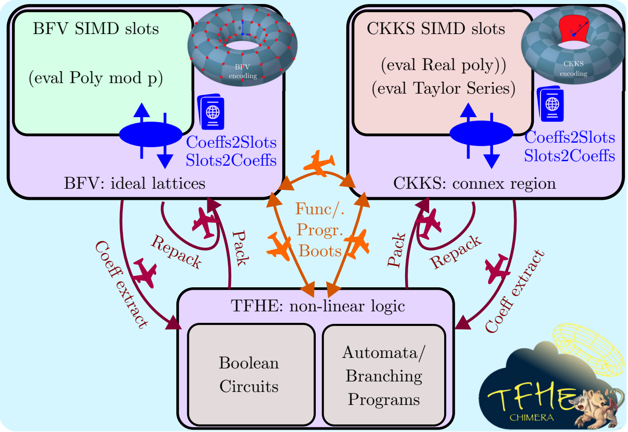
Strengths of FHE schemes

- BGV/BFV: SIMD integer arithmetic
- CKKS: SIMD fixed point arithmetic
- CGGI(TFHE)/DM: single evaluation, boolean logic, comparison, threshold, complex circuits
- etc...

How to get all the benefits without the limitations?

Scheme Swiching: Chimera [BGGJ20]

- Unified plaintext space
- Switch between ciphertext representations
- Implement bridges between TFHE, BFV and CKKS



Fully Homomorphic Schemes

inpher

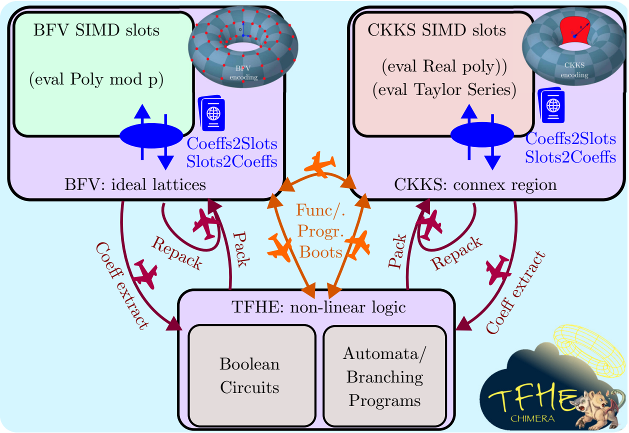
Strengths of FHE schemes

- BGV/BFV: SIMD integer arithmetic
- CKKS: SIMD fixed point arithmetic
- CGGI(TFHE)/DM: single evaluation, boolean logic, comparison, threshold, complex circuits
- etc...

How to get all the benefits without the limitations?

Scheme Swiching: Chimera [BGGJ20]

- Unified plaintext space
- Switch between ciphertext representations
- Implement bridges between TFHE, BFV and CKKS



Fully Homomorphic Schemes

inpher

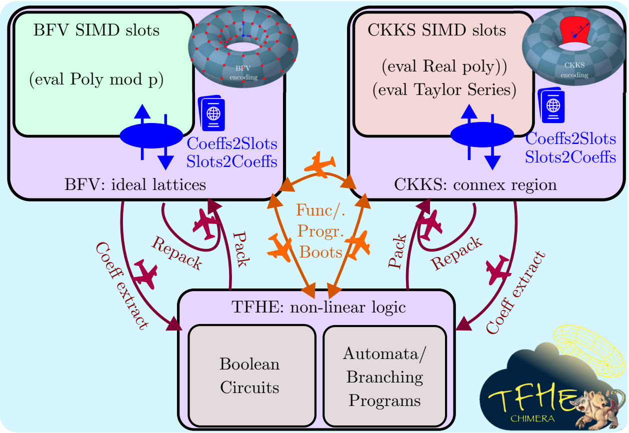
Strengths of FHE schemes

- BGV/BFV: SIMD integer arithmetic
- CKKS: SIMD fixed point arithmetic
- CGGI(TFHE)/DM: single evaluation, boolean logic, comparison, threshold, complex circuits
- etc...

How to get all the benefits without the limitations?

Scheme Swiching: Chimera [BGGJ20]

- Unified plaintext space
- Switch between ciphertext representations
- Implement bridges between TFHE, BFV and CKKS



FHE libraries

Library/ Scheme	BGV	BGV bootstr	BFV	CGGI'16	CGGI'17 advanced API	CKKS	CKKS bootstr	DM	Scheme switching
TFHE-lib				✓	*				*
TFHE-rs				✓	✓				
Lattigo	✓		✓			✓	✓		
HEAAN						✓	✓		
HELib	✓	✓				✓			
FHEW								✓	
OpenFHE	✓	*	✓	✓		✓	✓	✓	*
SEAL	✓		✓			✓			

* experimental

Coming soon! SPQlios-arithmetic: a middle-ground arithmetic API for FHE (included in TFHE-lib)

Supports **CRT and bivariate** frontends **at any depth** [BCGGJ23]: allowing efficient implementation of **Chimera**

- BlindRotate (CGGI bootstrapping in 6ms).
- CKKS and BFV products (depth 30 in 0.3s)
- Keyswitches and Automorphisms (depth 30 in 0.2s)

Running time is still bottleneck! → **target for hardware developers**!

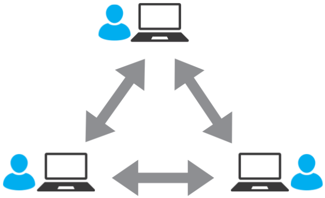
Secure Multi-party Computation

Secure Multi-party computation

inpher

Multi Party Computation

- Allows a set of parties to compute a function on their private data without revealing the inputs.
- Do this without putting all the data in the same room.
- Computation is enabled via interaction: thus **communication is the bottleneck**



Participants

- **Input party**: owns input data sources
- **Compute party**: performs the MPC computation
- **Output party**: receives the result
- Dealer (optional): generates the triplets/masks

Security Models

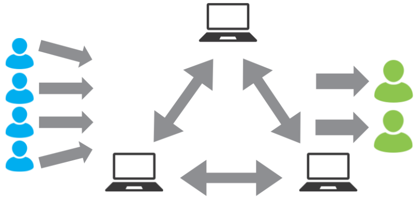
- **Passive** (honest-but-curious)–The players follow the protocol as prescribed.
- **Active** (malicious)– Attackers can make players deviate from the protocol.
- How many collusions are allowed? – **Honest** (nb coll$<$ half)/**Dishonest** majority (up to all except one)

Secure Multi-party computation

inpher

Multi Party Computation

- Allows a set of parties to compute a function on their private data without revealing the inputs.
- Do this without putting all the data in the same room.
- Computation is enabled via interaction: thus **communication is the bottleneck**



Participants

- **Input party**: owns input data sources
- **Compute party**: performs the MPC computation
- **Output party**: receives the result
- Dealer (optional): generates the triplets/masks

Security Models

- **Passive** (honest-but-curious)–The players follow the protocol as prescribed.
- **Active** (malicious)– Attackers can make players deviate from the protocol.
- How many collusions are allowed? – **Honest** (nb coll< half)/**Dishonest** majority (up to all except one)

Some building blocks

inpher

Linear Secret Sharing

- Additive Secret Sharing
- Shamir Secret Sharing
- Replicated Secret Sharing

Non-linear operations

- Mask and Reveal: e.g. Beaver triplets
- Encrypt and Reveal

Garbled Circuit

Yao's garbled circuit protocol allows a garbler to encode a boolean function into a garbled circuit that is evaluated by a second party, called the evaluator.

SMPC Schemes and Libraries

inpher

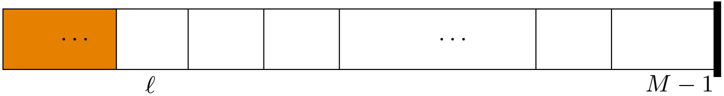
Library	Backend	Nb Players	Security Model
Manticore	A_{2^k}, B, Y	$n-$PC	passive with Dealer full-threshold
ABY/ABY3	A_{2^k}, B, Y	$2/3-$PC	passive, replicated SS honest majority
MP-SPDZ	$A_p,\ A_{2^k}, B, Y$	$n-$PC	active/passive, replicated SS/dealer/FHE honest/dishonest majority
Scala-Mamba	A_p, B, Y	$n-$PC	active honest/dishonest (A_p) majority
Sharemind	A_{2^k}, B, Y	$3-$PC	passive, replicated SS honest majority
FRESCO	$A_p,\ A_{2^k}, B$	$n-$PC	active (A), passive (B) dishonest majority
TinyGarble	Y	2-PC	passive
MPyC	A_p	n-PC	passive honest/dishonest (Shamir SS)

Computation domain (backend) : a range $[0, L-1]$
$L = 2$ boolean backend (B)
$L > 2$ arithmetic backend (A)
if L a power of 2: "native" arithmetic (A_{2^k})
if L a prime number: "field" arithmetic (A_p)
Garbled circuits: (Y)

Zoom into Manticore [Belorgey et al.'23]

inpher

$$\mathcal{M}_{M,\ell} = \sum_{i=\ell}^{M-1} m_i 2^i \quad \text{mod } 2^M, \quad \text{for} \quad m_i \in \{0,1\}$$



ModReal representation

- Numerical window in Manticore is not fixed, so we can increase or decrease M or ℓ
- thus improving on the sizes of the representation as well as the communication cost
- Allows for automating the estimate of these parameters at compile time
- Real number arithmeric with high numerical precision (≥ 7 digits after the decimal point)

Lift algorithm

- Uses masking data precomputed in the setup phase and is without error probability.
- Compared to [Escudero et al.'20] (logical right shift used in MP-SPDZ) provides better efficiency by avoiding the use of 2 oblivious comparisons

SMPC Logistic regression benchmarks

inpher

Dataset	Method	Log-loss	Exec. time (sec)	Comm. (MB)
X	Manticore	**0.445**	12.4	512
	mp-spdz a	0.449	5.5	263
	mp-spdz b	0.449	35.4	1020
	mp-spdz c	**0.445**	43.4	2697
	mp-spdz d	**0.445**	414.9	25352
X'	Manticore	**0.445**	12.5	512
	mp-spdz a	8.549	5.5	263
	mp-spdz b	4.689	35.6	1020
	mp-spdz c	17.102	43.5	2697
	mp-spdz d	3.821	415.8	25352
$X\|X$	Manticore	**0.445**	12.8	539
	mp-spdz a	0.652	5.5	264
	mp-spdz b	0.695	36.3	1033
	mp-spdz c	0.825	43.6	2698
	mp-spdz d	0.819	415.2	25365

`mp-spdz a` uses a 5 piece-wise sigmoid approx. with 10 iterations, `mp-spdz b` uses a 5 piece-wise sigmoid approx. with 100 iterations, `mp-spdz c` uses exact sigmoid with 10 iterations, `mp-spdz d` uses exact sigmoid with 100 iterations.

- MP-SPDZ:
  - Mini-batch logreg (batch size 128)
  - Using edaBits and $\mod 2^k$ plaintext space
  - Fixed-point parameters 16.16 (could not compile with higher decimal precision)
  - Honest majority, 3 parties, non-malicious, replicated secret-sharing
- Manticore:
  - Full threshold, 3 parties, non-malicious
  - 10 IRLS + 2 SGD iterations

Datasets $(30k \times 10)$

- X – random full-rank in the interval $[-4, 4]$
- X' – re-scaled a column of X by 2^8
- $X|X$ – correlated features

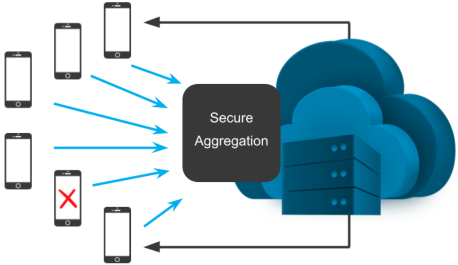
Federated Learning and Differential Privacy

Federated Learning

Problem: train a deep neural network on horizontally split data (same features) across multiple client devices

Federated Learning as an Edge Computing Framework

- FL is NOT a privacy-preserving method, it is a framework combining different PPTs (MPC, FHE, DP etc.) to support massively distributed ML computations



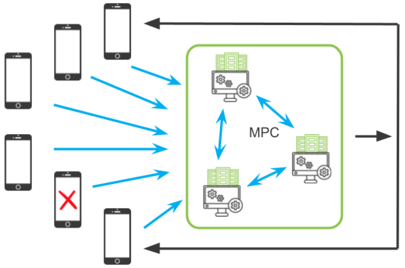
Challenges

- Millions of clients with limited compute capabilities
- Complex models (millions/billions of coefficients)
- Several TB of model coefficients to communicate
- Unstable connectivity of the clients (drop-out)
- Synchronization of communication
- Need for Adapted Optimization:
  not just linear operations

Secure Aggregation

inpher

Leaking input data from client to server via model updates (inversion attacks)



Secure aggregation protocols (Single-server or Multi-servers setting)

- Differential privacy
- Pairwise additive masking
- Threshold (fully) homomorphic encryption
- Secure multiparty computation
- Trusted execution environments

[BDGJM23] Falkor: Federated Learning Secure Aggregation Powered by AES-CTR GPU Implementation

Differential Privacy

inpher

Goal:

- Output does not reveal whether an individual was in the input database $\rightarrow$ output privacy
- Adds small amounts of randomness to a dataset, a model, or an output to protect individual samples

Callenges

- More noise yields better privacy but also degrades the utility of the result.
- However, every query on the underlying private data results in some amount of information being revealed.
- Given enough computations or queries on the same data, an attacker might be able to learn about the input.
- Each application therefore needs a privacy budget.

Privacy Budget

Indicates how much information is allowed to be revealed cumulatively across all queries/computations.

Outcome on PETs

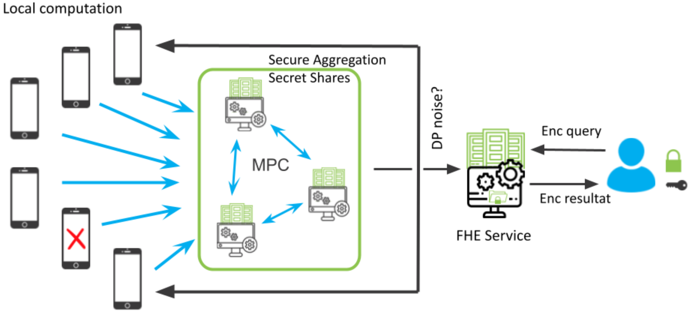
Which privacy preserving technology will be the best?

inpher

	FHE	MPC	secure FL	plaintext + DP FL
data stacking	any	any	vertical only	vertical only
performance	compute-bound	network-bound	medium	fast
hardware req.	large/specific	medium to large	normal/light	normal/light
conv. speed	fast	fast	medium	slow
security	encryption	secret sharing/non-collusion	agg. reveal	individual grad. leakage
interactive	non	yes	yes	yes
precision	medium/high	high	medium	low

Chaining heterogeneous computations

- In 2024, nobody believes anymore that there exists single best technology.
- Most of the use-cases require to compose many PPC/PET technology out of MPC, FHE, DP, TEE ...
- Chaining heterogeneous PPC computation to control the privacy of the data flow.



Challenges

- Each input under which visibility: plaintext, shares, or ciphertexts?
- Transfers of visibility/ ownership legitimate? (GDPR, ethical, etc...)?
- What are the variables allowed to reveal?
- Do we need DP noise?
- What are the allowed computations?
- Privacy Budget?

Standardization

inpher

- ISO/IEC AWI 28033 Fully homomorphic encryption (under development)
  Part 1: General
  Part 2: Mechanisms for exact arithmetic on modular integers (BFV/BGV)
  Part 3: Mechanisms for arithmetic on approximate numbers (CKKS)
  Part 4: Mechanisms for arithmetic based on evaluation of digital circuits, look-up tables and deterministic automata (CGGI/DM)
  Part 5: Mechanisms for Scheme Switching (Chimera)
- ISO/IEC 4922-1:2023: Part 1: General (definitions, terminology and processes for MPC) (published)
- ISO/IEC 4922-2:2024 Part 2: Mechanisms based on secret sharing (published)
- ISO/IEC 4922-3 Part 3: Mechanisms based on garbled circuits (under development)
- PWI 24836 Oblivious Transfer (under development)
- NIST's Multi-Party Threshold Cryptography standardization project
- NIST's Guidelines for Evaluating Differential Privacy Guarantees (initial public draft)

GenoPPML
Genomic Privacy-Preserving Machine Learning

Introduction

inpher

Goal of GenoPPML framework

Secure machine learning on genomic data

Problem

- Train a machine learning model on gene data, where data is owned by ≥ 2 parties.
  - breast tumors prediction, cancer relapse, tumor differentiation, etc.
- DP federated learning for cancer prediction model.
- Privacy preserving predictions keeping data private and model secure.

Solution [CGGJ22]

- MPC for learning a breast cancer prediction model (Manticore)
- DP for protecting the revealed model
- HE for predicting on encrypted data (TFHE)
- Idash 2020 – secure genome analysis competition

Introduction

inpher

Goal of GenoPPML framework

Secure machine learning on genomic data

Problem

- Train a machine learning model on gene data, where data is owned by ≥ 2 parties.
  - breast tumors prediction, cancer relapse, tumor differentiation, etc.
- DP federated learning for cancer prediction model.
- Privacy preserving predictions keeping data private and model secure.

Solution [CGGJ22]

- MPC for learning a breast cancer prediction model (Manticore)
- DP for protecting the revealed model
- HE for predicting on encrypted data (TFHE)
- Idash 2020 – secure genome analysis competition

Introduction

inpher

Goal of GenoPPML framework

Secure machine learning on genomic data

Problem

- Train a machine learning model on gene data, where data is owned by ≥ 2 parties.
  - breast tumors prediction, cancer relapse, tumor differentiation, etc.
- DP federated learning for cancer prediction model.
- Privacy preserving predictions keeping data private and model secure.

Solution [CGGJ22]

- MPC for learning a breast cancer prediction model (Manticore)
- DP for protecting the revealed model
- HE for predicting on encrypted data (TFHE)
- Idash 2020 – secure genome analysis competition

General overview

inpher

data owners
(data remains local)

X_A
y_A

X_B
y_B

Analyst

final users

General overview

data owners
(data remains local)

X_A
y_A

local to A

Local Processing

X_B
y_B

local to B

Local Processing

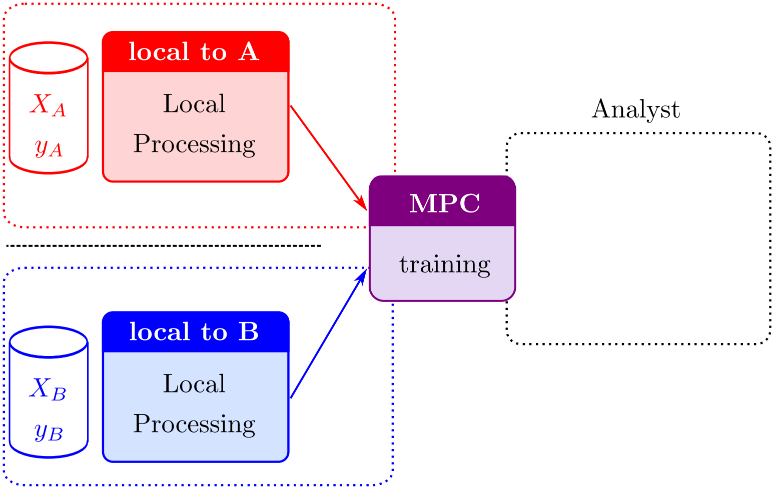
Analyst

final users

General overview

data owners
(data remains local)

final users



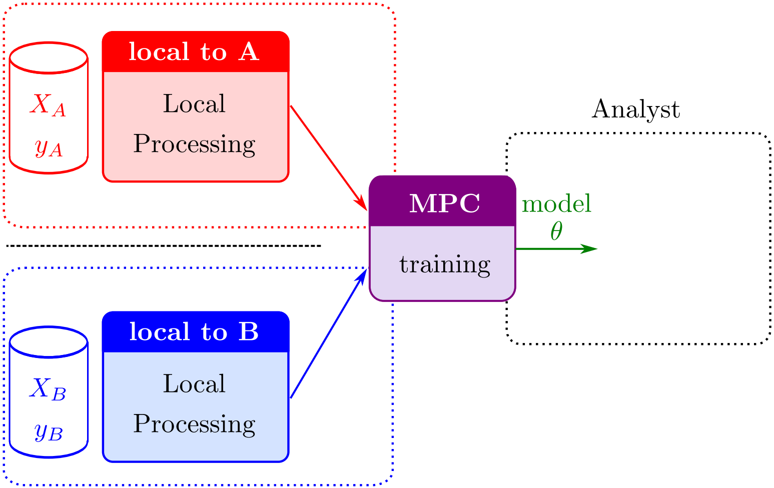
General overview

inpher

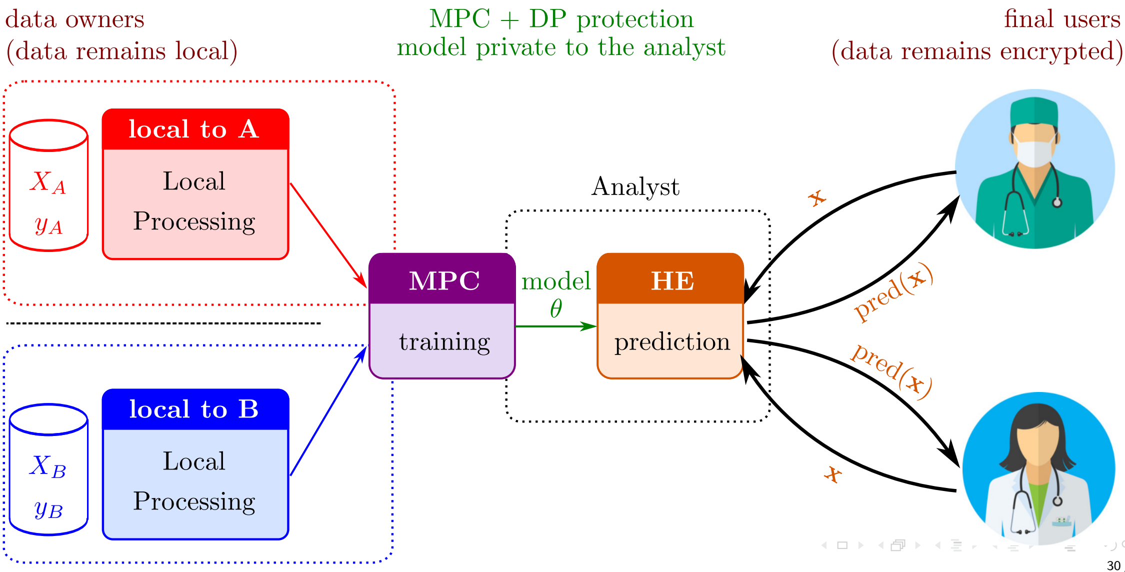
data owners
(data remains local)

MPC + DP protection
model private to the analyst

final users

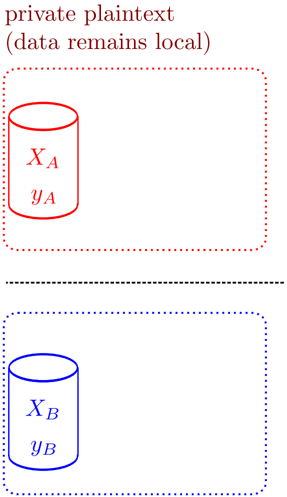


General overview



Implementation details

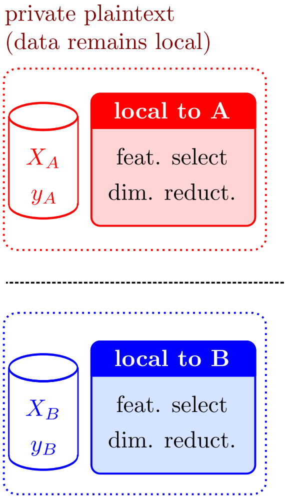
inpher



1. Each data owner first execute a local computation (feature selection)
2. Both perform a common secret-shared MPC computation (logistic regression)
3. Add noise around the baseline model during the MPC computation (for DP)
4. Reveal DP protected model

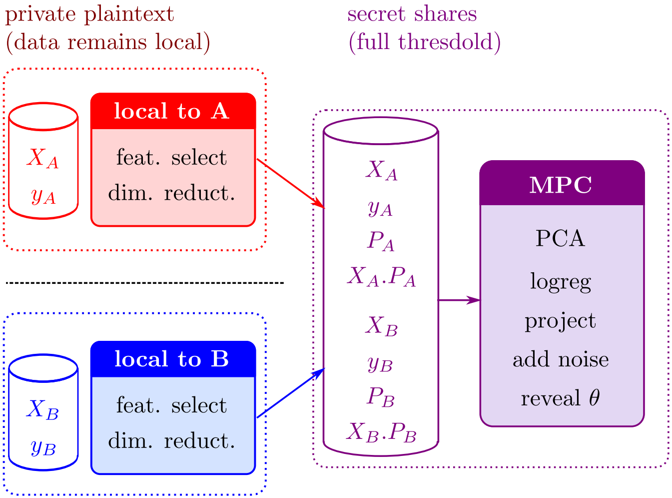
Implementation details

inpher



1. Each data owner first execute a local computation (feature selection)
2. Both perform a common secret-shared MPC computation (logistic regression)
3. Add noise around the baseline model during the MPC computation (for DP)
4. Reveal DP protected model

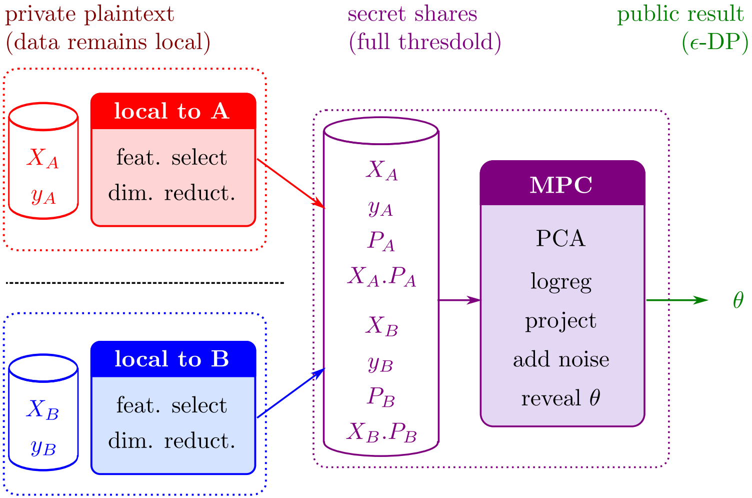
Implementation details



1. Each data owner first execute a local computation (feature selection)
2. Both perform a common secret-shared MPC computation (logistic regression)
3. Add noise around the baseline model during the MPC computation (for DP)
4. Reveal DP protected model

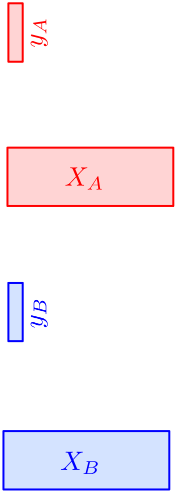
Implementation details

inpher



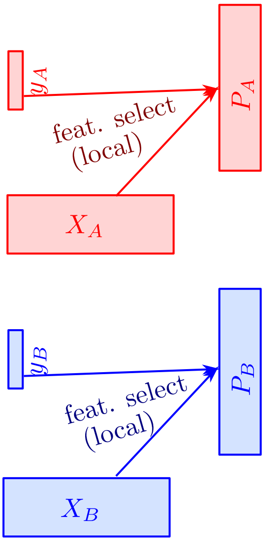
1. Each data owner first execute a local computation (feature selection)
2. Both perform a common secret-shared MPC computation (logistic regression)
3. Add noise around the baseline model during the MPC computation (for DP)
4. Reveal DP protected model

Training algorithm



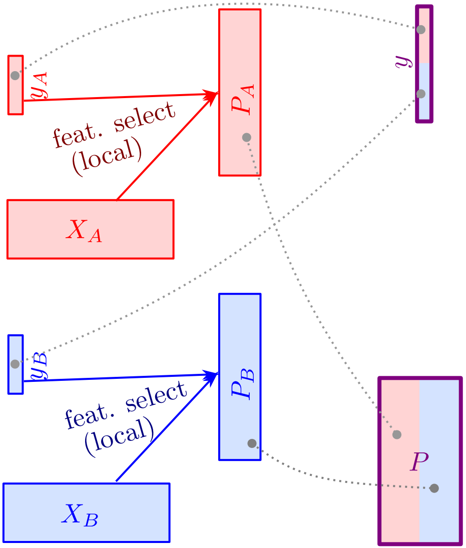
Training algorithm

inpher

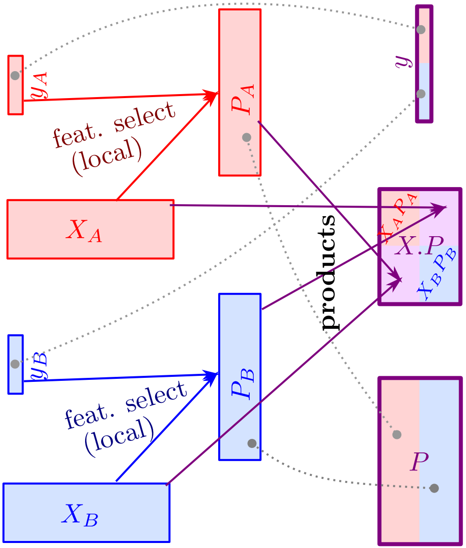


Training algorithm

inpher

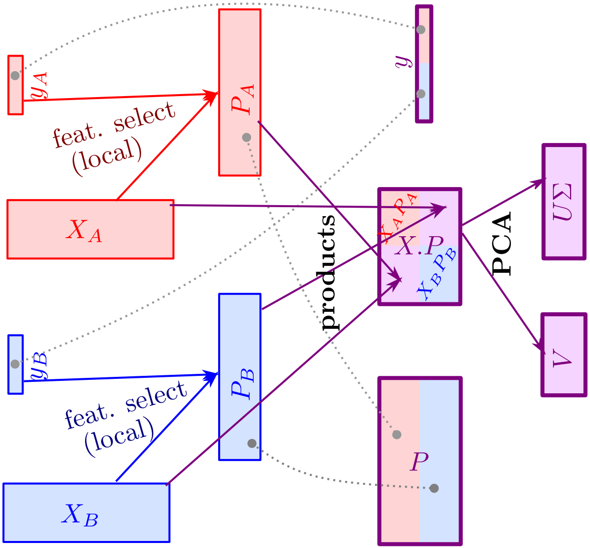


Training algorithm



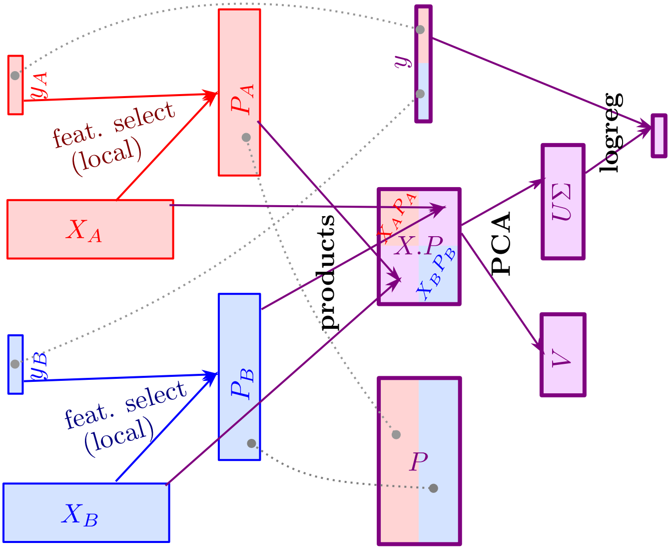
Training algorithm

inpher



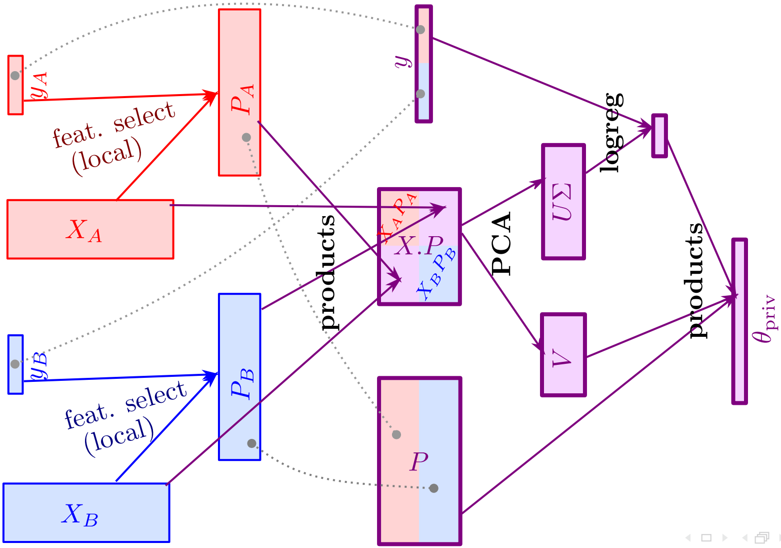
Training algorithm

inpher

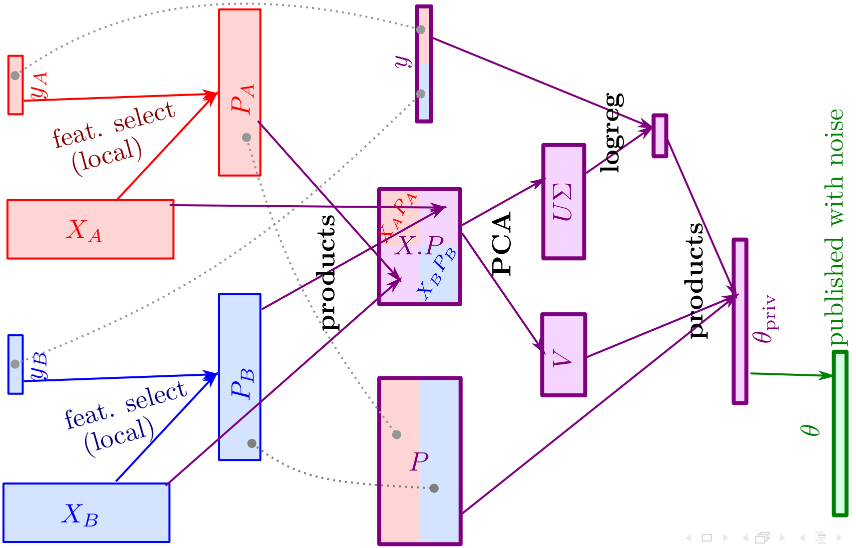


Training algorithm

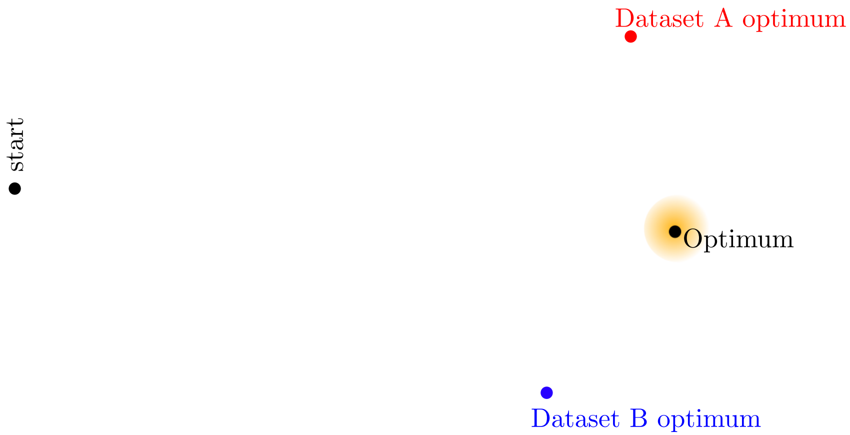
inpher



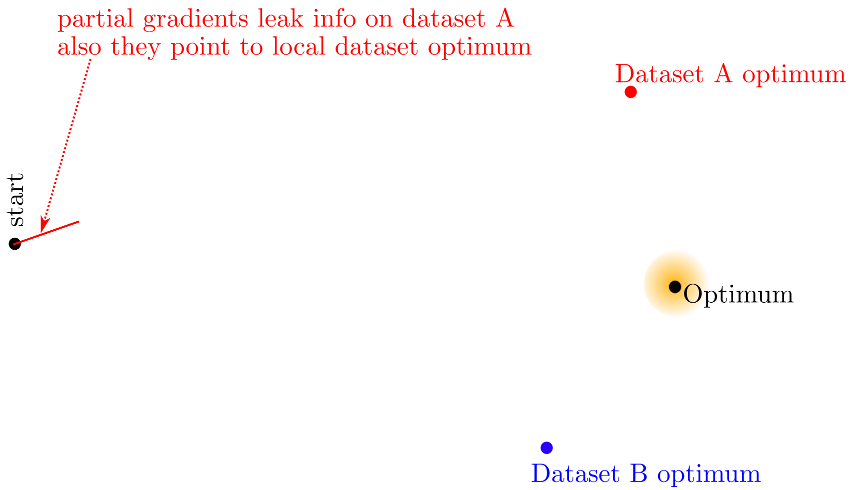
Training algorithm



MPC versus plain Federated Learning

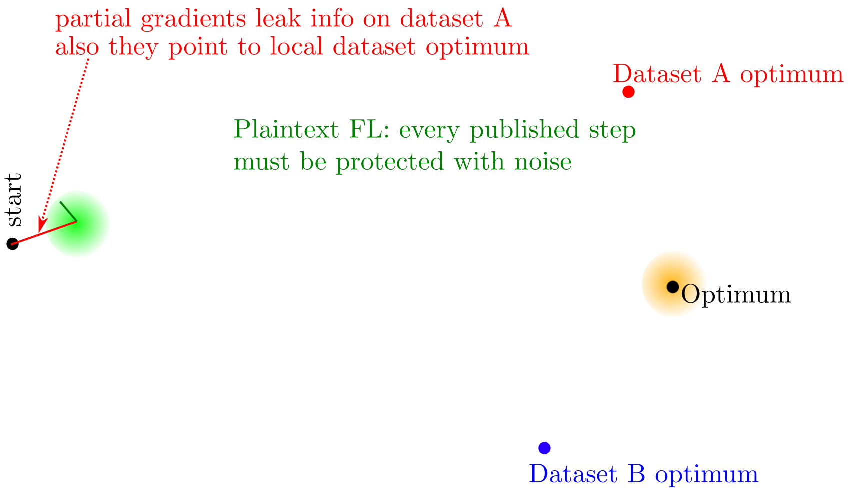


MPC versus plain Federated Learning



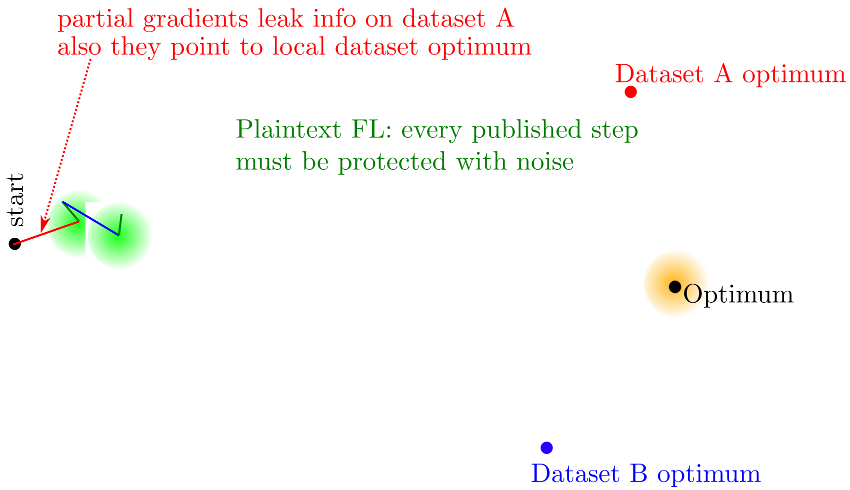
MPC versus plain Federated Learning

inpher



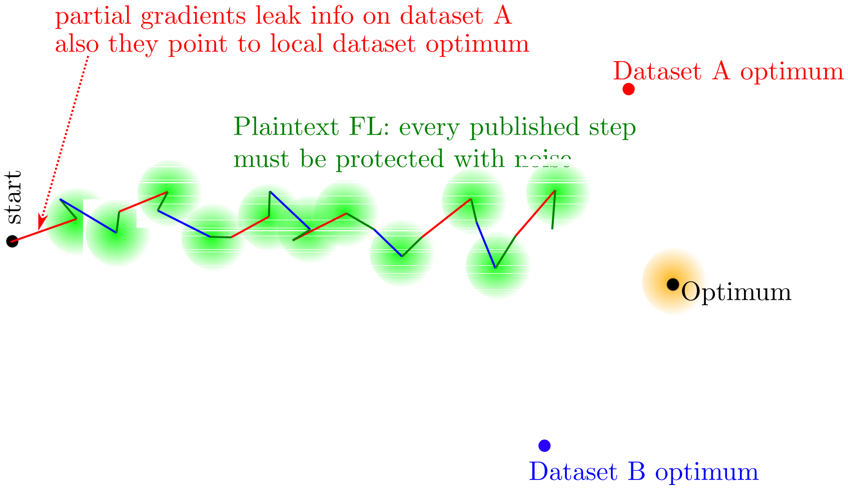
MPC versus plain Federated Learning

inpher



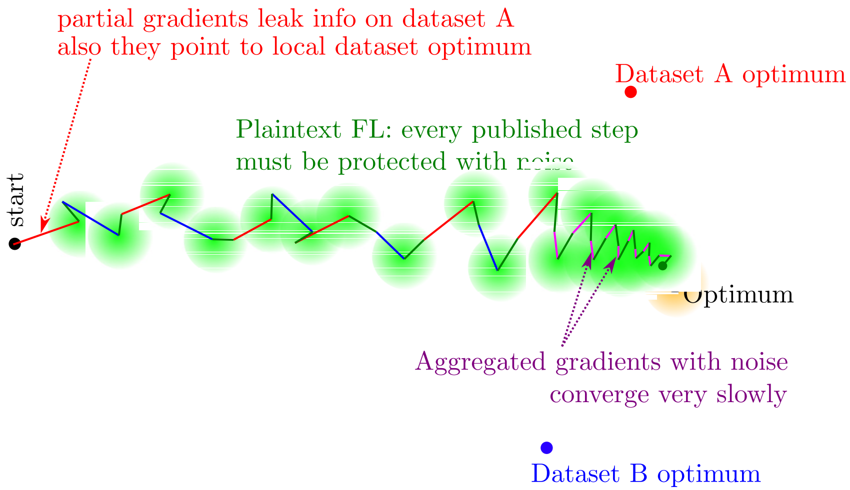
MPC versus plain Federated Learning

inpher

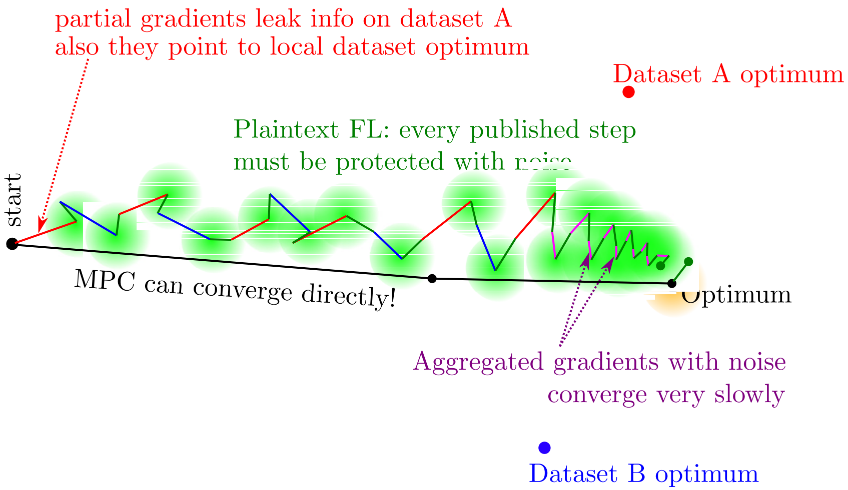


MPC versus plain Federated Learning

inpher



MPC versus plain Federated Learning



Main advantages of GenoPPML

inpher

Feature selection

- Each party implements its own feature selection which stays private
- Reduces dimensionality of dataset, speeding-up MPC step

Secret-shared MPC using Manticore

- No temporary variable, or partial gradient is published
- No restriction on the choice of aggregation function:
  - faster convergence method (IRLS): converge in 8-10 iterations
  - reduced number of communications rounds
- Operates over the full dataset
  - stable even if the datasets of player A and B are not i.i.d.

Main advantages of GenoPPML

inpher

Feature selection

- Each party implements its own feature selection which stays private
- Reduces dimensionality of dataset, speeding-up MPC step

Secret-shared MPC using Manticore

- No temporary variable, or partial gradient is published
- No restriction on the choice of aggregation function:
  - faster convergence method (IRLS): converge in 8-10 iterations
  - reduced number of communications rounds
- Operates over the full dataset
  - stable even if the datasets of player A and B are not i.i.d.

Numerical stability and DP Noise

inpher

Numerical stability

- L2 regularization of logreg
- Less over-fitting
- PCA dimension reduction inside MPC
- Mitigates influence of individual samples
  - i.e. less DP noise required

Where to add DP noise in projected logreg?

- No individual gradients leaked $\Longrightarrow$ no noise here!
- Only final model is published $\Longrightarrow$ one-time DP noise
- Supports ϵ and (ϵ, δ) DP

Numerical stability and DP Noise

inpher

Numerical stability

- L2 regularization of logreg
- Less over-fitting
- PCA dimension reduction inside MPC
- Mitigates influence of individual samples
  - i.e. less DP noise required

Where to add DP noise in projected logreg?

- No individual gradients leaked $\Longrightarrow$ no noise here!
- Only final model is published $\Longrightarrow$ one-time DP noise
- Supports ϵ and (ϵ, δ) DP

Experiments

TABLE I
TRAIN AND TEST DATASET SIZES TOGETHER WITH POSITIVE TO NEGATIVE CLASS RATIO.

	#features	#samples train	#samples test	neg. to pos. ratio
BC-TCGA	17,814	471	119	0.12
GSE2034	12,634	228	58	0.60
BC12-TCGA	25,128	2,169	544	0.08

Accuracy w.r.t. DP-noise

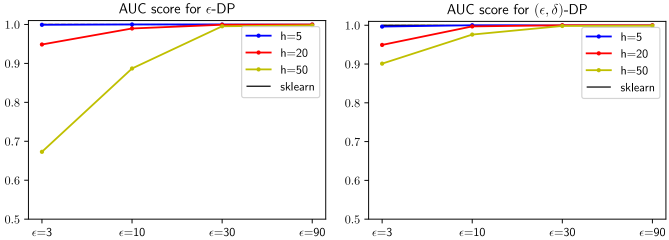


Fig. 3. BC-TCGA dataset AUC scores.

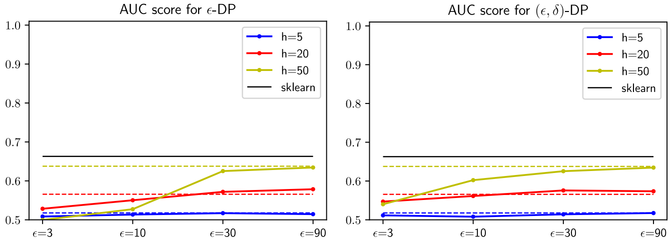


Fig. 4. GSE2034 dataset AUC scores.

Accuracy w.r.t. DP-noise

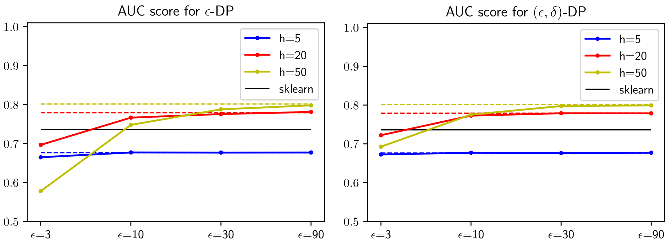
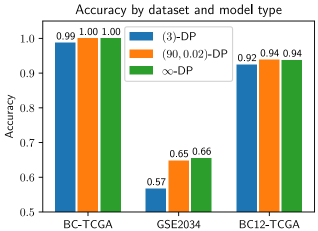


Fig. 5. BC12-TCGA dataset AUC scores.



Best obtained model accuracy by dataset and model type.

Benchmarks (Training Phase)

inpher

TABLE II
EXECUTION TIMES (IN SECONDS PER PLAYER), RAM USAGE AND NETWORK COMMUNICATION (IN MB PER PLAYER).

Dataset		h	5	20	50
BC-TCGA	Local processing	CPU	0.49	0.99	1.78
		RAM	266	267	270
	MPC	CPU	0.91	1.30	2.17
		RAM	399	437	466
		Network	138	164	219
GSE2034	Local processing	CPU	0.12	0.29	0.51
		RAM	161	161	161
	MPC	CPU	0.37	0.56	1.02
		RAM	167	180	276
		Network	50	68	106
BC12-TCGA	Local processing	CPU	4.10	6.30	11.31
		RAM	1294	1294	1294
	MPC	CPU	5.87	7.43	11.47
		RAM	1892	1925	2005
		Network	854	909	1021

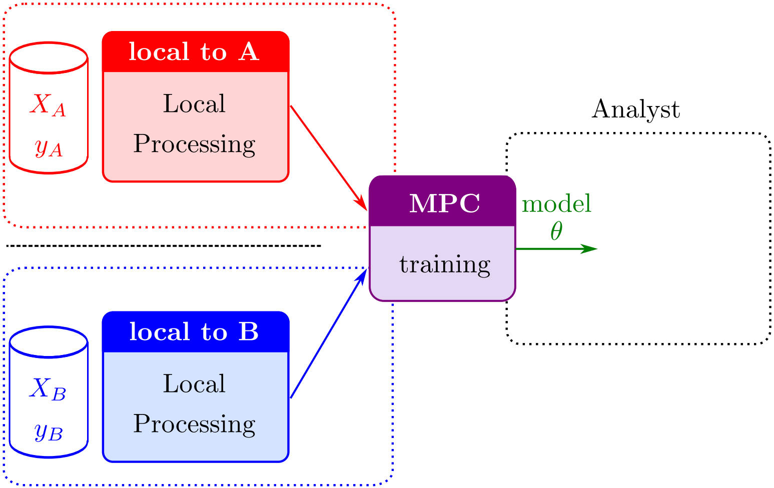
FHE predictions

inpher

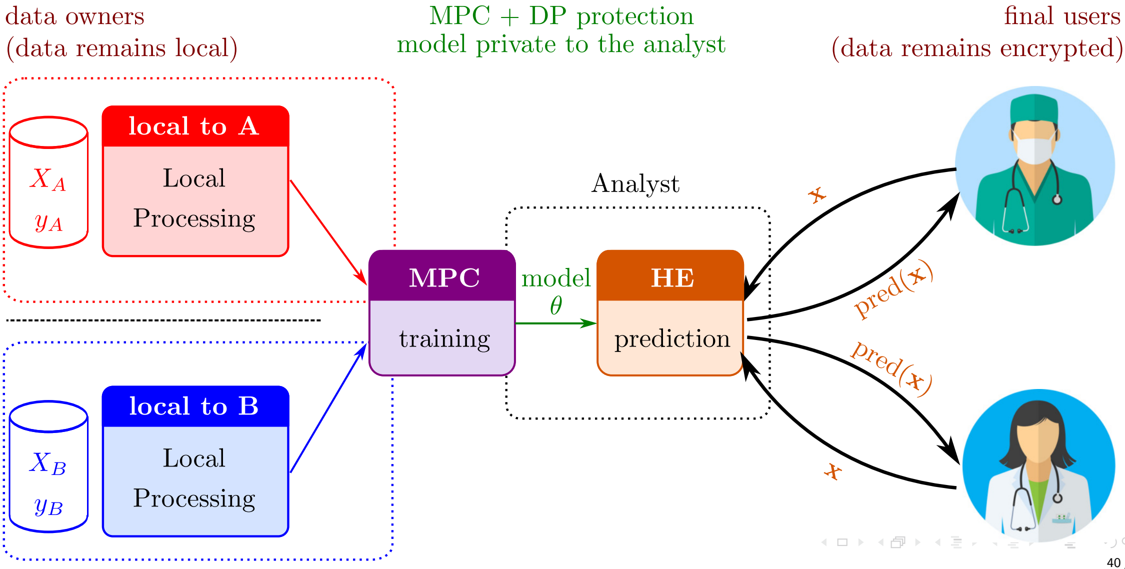
data owners
(data remains local)

MPC + DP protection
model private to the analyst

final users



FHE predictions



FHE predictions

inpher

TFHE library

- Open source FHE library `https://tfhe.github.io/tfhe`
- C/C++ distributed under Apache 2.0 license
- 128 bits of security (binary secrets) or 176 bits with ternary secrets

Building blocks

- Logreg is plaintext (logreg model) $\times$ ciphertext (user data) dot product and a sigmoid evaluation
- Plaintext $\times$ ciphertext dot product:
  - TRLWE multiplication with IntPolynomial + TLWE Coeff Extract
- Sigmoid/Sign evaluation:
  - Programmable bootstrapping (blind rotate)

FHE predictions

TFHE library

- Open source FHE library `https://tfhe.github.io/tfhe`
- C/C++ distributed under Apache 2.0 license
- 128 bits of security (binary secrets) or 176 bits with ternary secrets

Building blocks

- Logreg is plaintext (logreg model) $\times$ ciphertext (user data) dot product and a sigmoid evaluation
- Plaintext $\times$ ciphertext dot product:
  - TRLWE multiplication with IntPolynomial + TLWE Coeff Extract
- Sigmoid/Sign evaluation:
  - Programmable bootstrapping (blind rotate)

FHE Predictions – benchmarks

inpher

Space requirements

- Keys:
  - Secret key – 128B
  - Public key – 48MB
- Query:
  - BC-TCGA – 144kB (18 TRLWE) for 18k features
  - GSE2034 – 104kB (13 TRLWE) for 12k features
  - BC12-TCGA – 200kB (25 TRLWE) for 25k features
- Result size:
  - 1 TLWE ciphertext – 4kB

Timings per query

- Encrypt/Decrypt $\approx$ 10ms
- Logreg prediction $\approx$ 90ms

FHE Predictions – benchmarks

inpher

Space requirements

- Keys:
  - Secret key – 128B
  - Public key – 48MB
- Query:
  - BC-TCGA – 144kB (18 TRLWE) for 18k features
  - GSE2034 – 104kB (13 TRLWE) for 12k features
  - BC12-TCGA – 200kB (25 TRLWE) for 25k features
- Result size:
  - 1 TLWE ciphertext – 4kB

Timings per query

- Encrypt/Decrypt $\approx$ 10ms
- Logreg prediction $\approx$ 90ms


Eurocrypt 2024 Affiliated Event
May 26, 2024
Zurich, Switzerland
Tutorial & Practices on Hybrid Pets
Hybrid PETs
TUTORIALS
Cat or Dog? What PETS Are and How to Choose Them
Nigel Smart (KU Leuven, Zama)
Introduction to FHE and CKKS performance improvements
Damien Stehle (CryptoLab)
Introduction to SMPC and hybrid privacy preserving applications
Mariya Georgieva, Sergiu Carpov (Inpher)
PRACTICAL SESSIONS
Machine learning workflows using Inpher's XOR Platform
Marc Desgroseilliers (Inpher)
New FFT and arithmetic API for Fully Homomorphic Encryption Libraries
Nicolas Gama (SandboxAQ, Inpher)
Confidential smart contracts using threshold FHE and the Zama fhEVM
Morten Dahl (Zama)

Thank you

inpher

References

- [Escudero et al'20] D. Escudero, S. Ghosh, M. Keller, R. Rachuri, P. Scholl, Improved primitives for MPC over mixed arithmetic-binary circuits
- [BGGJ20] C. Boura, N. Gama, M. Georgieva, D. Jetchev: CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes.
- [CGGJ22] S. Carpov and N. Gama and M. Georgieva and D. Jetchev: GenoPPML – a framework for genomic privacy-preserving machine learning.
- [Belorgey et al'23] M. Belorgey, S. Carpov, K. Deforth, N. Gama, D. Jetchev, J. Katz, I. Leontiadis, M. Mohammadi, A. Sae-Tang, M. Vuille: Manticore: A Framework for Efficient Multiparty Computation Supporting Real Number and Boolean Arithmetic.
- [BDGJM23] Mariya Georgieva Belorgey, Sofia Dandjee, Nicolas Gama, Dimitar Jetchev, Dmitry Mikushin: Falkor: Federated Learning Secure Aggregation Powered by AES-CTR GPU Implementation
- [BCGGJ23] M.G. Belorgey, S. Carpov, N. Gama, S. Guasch, D. Jetchev: Revisiting Key Decomposition Techniques for FHE: Simpler, Faster and More Generic.