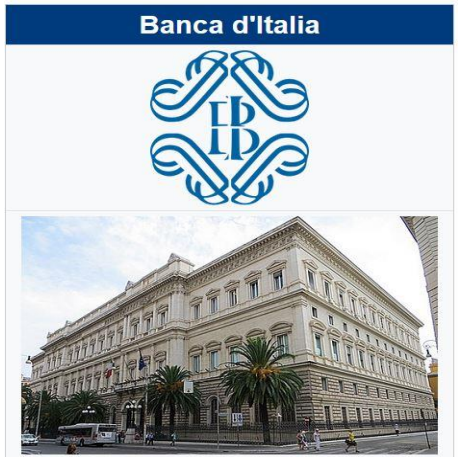
Data sharing conundrum? In fully homomorphic encryption we (must) trust

Presenter: Adriano Baldeschi PhD

Authors: Adriano Baldeschi, Giuseppe Bruno, Mirko Avantaggiato, Andrea Capitanelli



CORNAMI®
Intelligent Computing

What is Cryptography?

- Definition:
- - Cryptography is a technique to protect information by converting it into an unreadable format without a secret key.

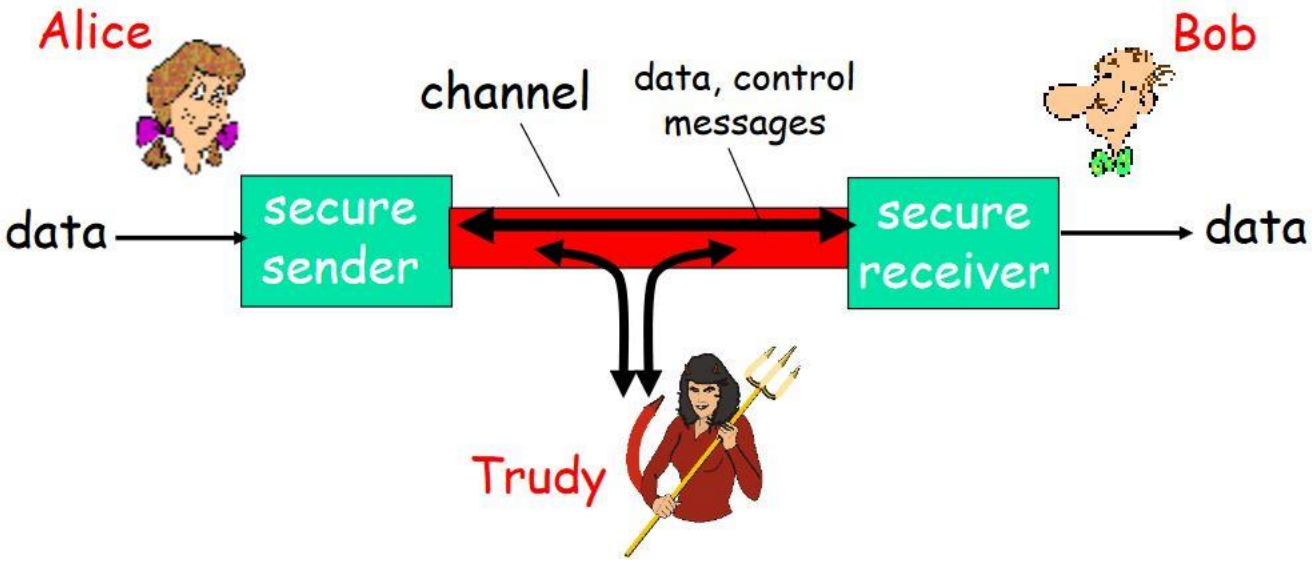
- Main Objectives:
- 1. Confidentiality
- 2. Integrity

A little bit of naming

A cryptographic system is a system capable of encrypting and decrypting a message through the use of an algorithm and a key (an alphanumeric string). The message to be encrypted is called “plaintext” while the result of the cryptographic algorithm is called “ciphertext”

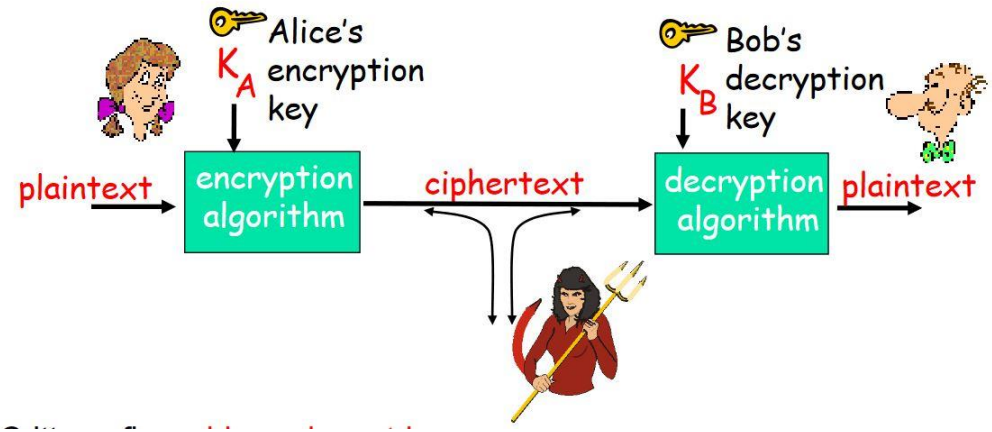
Alice and Bob are the two parties that intend to communicate securely through the network

- They could be client and server programs, or devices (e.g. routers) …
- Trudy is a third party that can listen to the messages exchanged by Alice and Bob and possibly alter them, delete them or create fake ones



Types of Cryptography

- Main Categories:
- - Symmetric Cryptography: One key for both encryption and decryption.
- - Asymmetric Cryptography: A pair of public and private keys.



Introduction to Homomorphic Encryption

- What is it?
- - A type of encryption that allows computations on encrypted data without decrypting it.

- Key Feature:
- - The computation result remains encrypted but, when decrypted, matches the result of working on plaintext data.

- Origin of the term:
- - 'Homomorphic' means 'same form' in Greek.

How Does Homomorphic Encryption Work?

- 1. Encryption: Data is transformed into encrypted form.
- 2. Computation on encrypted data: Operations like addition and multiplication are performed without access to plaintext.
- 3. Decryption: The encrypted result is transformed into plaintext, providing the correct output.

- Note: Computationally intensive but increasingly practical.

An example

Scenario: Credit Risk Analysis with Homomorphic Encryption

A bank wants to calculate a customer's **creditworthiness** based on their **income** and **total debt**, but without ever seeing the raw data for privacy reasons.

Problem:

- The customer does not want to share their income and debt in plaintext with the bank.
- The bank needs to compute the **debt-to-income ratio** to assess risk.
- If traditional encryption were used, the bank would not be able to perform calculations on the encrypted data.

Solution with Homomorphic Encryption

1. The customer **encrypts** their financial data with a homomorphic public key.
2. They send the encrypted data to the bank.
3. The bank performs computations directly on the encrypted data without decrypting it.
4. The encrypted result is returned to the customer.
5. The customer **decrypts** the result with their private key and obtains their creditworthiness score.

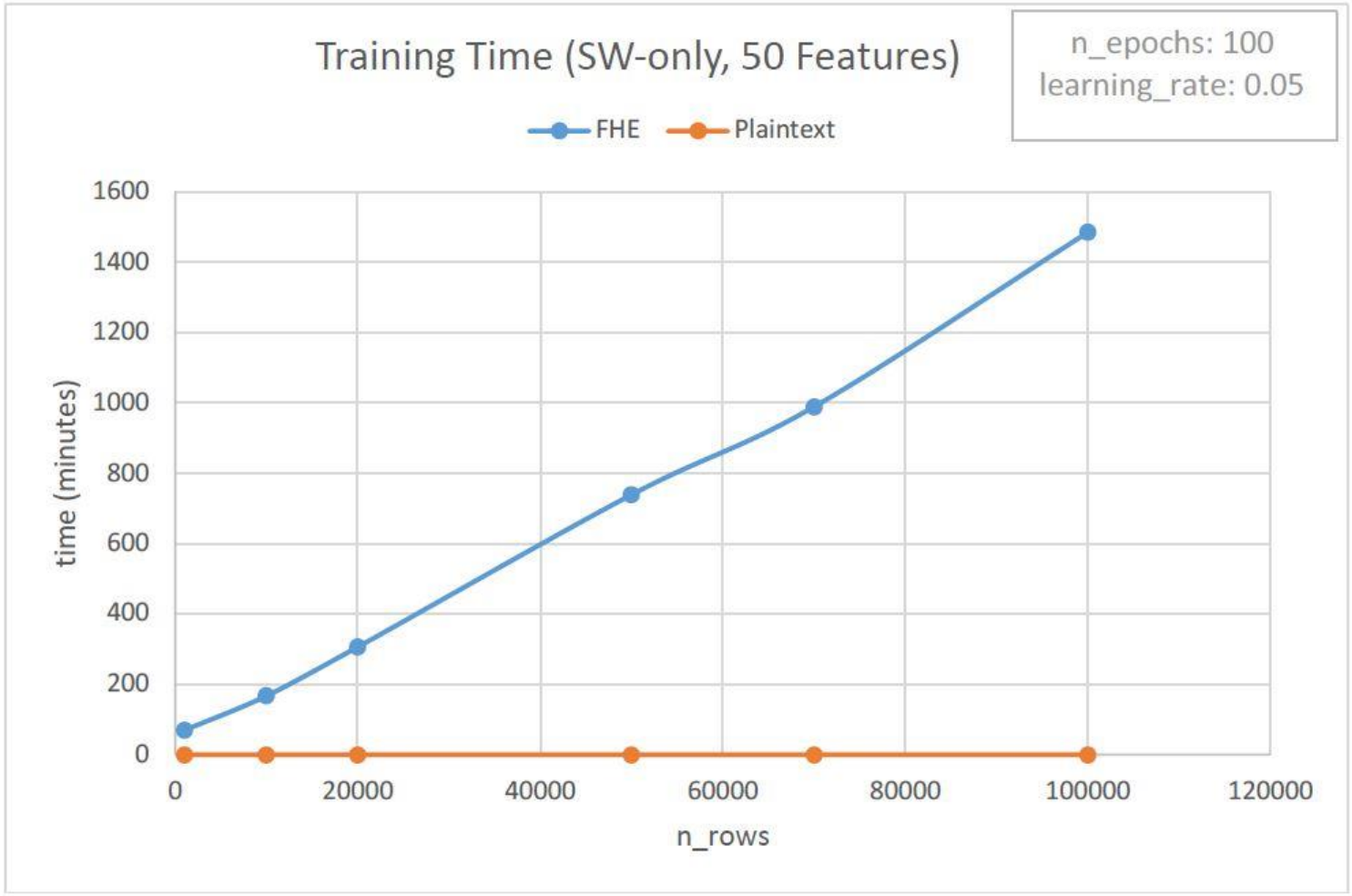
Using homomorphic encryption on linear regression

1. **Scenario**:
   - A healthcare organization wants to use patient data (e.g., age, weight, blood pressure) to train a linear regression model to predict disease risk.
   - The data is sensitive and must remain private throughout the analysis.
2. **Homomorphic Encryption in Action**:
   - The organization encrypts the dataset using a **homomorphic encryption scheme** before sharing it with a data analyst or external machine learning provider.
   - This encryption allows computations directly on the encrypted data without needing decryption.
3. **Steps in the Process**:
   - **Data Encryption**:
     Each data point (features and target values) is encrypted using a public key. For example:
     - X (features) → $Enc(X)$
     - y (target) → $Enc(y)$
   - **Secure Computation**:
     The analyst performs the required computations (e.g., matrix multiplication for gradient descent) directly on the encrypted data:
     - Compute $Enc(\beta) = (X^T X)^{-1} X^T y$, where β is the coefficient vector of the regression model.

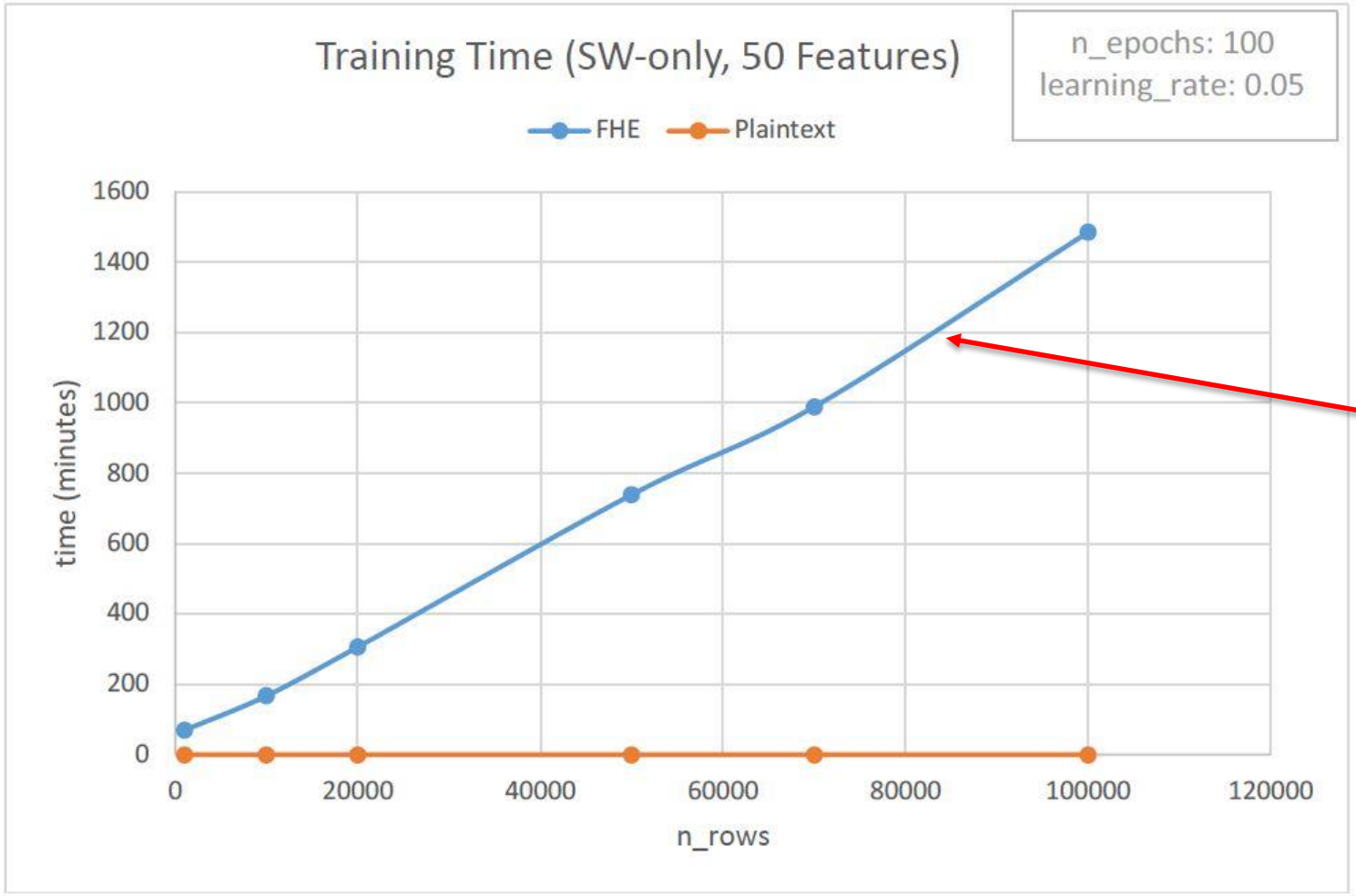
The gradient descent approach to linear regression

The gradient descent approach allows us to fit a linear regression model on a variety of datasets. In particular, we focus on datasets that can have up to 50 columns for independent variables (features) and up to 100000 rows (one hundred thousand rows). To measure the software-only (SW-only) performance results we selected various datasets with different numbers of rows and columns. Specifically, we ran several experiments by fixing the number of columns to five, thirty and fifty. For each case, we used different number of rows and measured the run time for fitting a linear regression model and making inference (prediction) with the trained model. The experimental results are as shown in the following tables and plots.

Training time



Training time



Slow!

Training time on Cornami accelerated Hardware

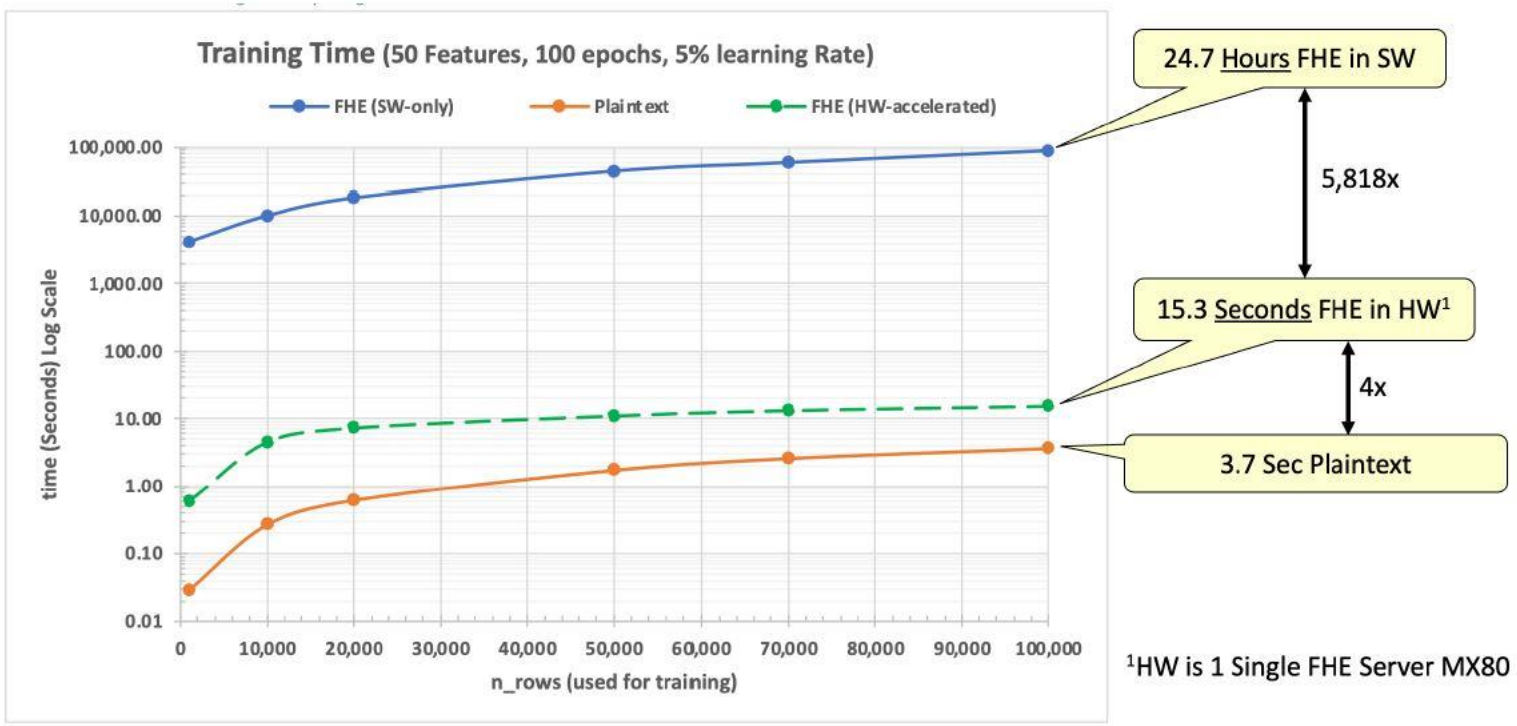


Figure 9: Training Runtime for Plaintext, FHE SW-only, Cornami FHE HW

Statistical Analysis of the Results

RMSE: root mean square error

n_rows	n_cols	RMSE (FHE and actual values)	RMSE (PTXT and actual values)
250	50	0.0064	0.00626
2500	50	0.00535	0.00499
5000	50	0.00529	0.00486
12500	50	0.0065	0.00618
17500	50	0.00683	0.00658
25000	50	0.00567	0.00533

When n_rows>1000 we obtain good results

Conclusion

- We trained a linear regression model on fully homomorphic encrypted (FHE) data using the stochastic gradient descent algorithm.
- The estimated coefficients are comparable to the actual ones when the dataset contains more than 1,000 rows.
- Although training time is relatively slow using software, it can be significantly faster using Cornami hardware.