Imperial College London

Ethics of Data Science – Part III

Week 3: Homomorphic Encryption

Polynomial ML - Deep learning

Dr. Chris Anagnostopoulos, Hon. Senior Lecturer

FHE: the two questions that matter

- Which exact part(s) of the ML pipeline do you want to protect, and from whose perspective?
- What type of computations are actually feasible using FHE?

There are many other interesting research questions, but practically these two are critical.

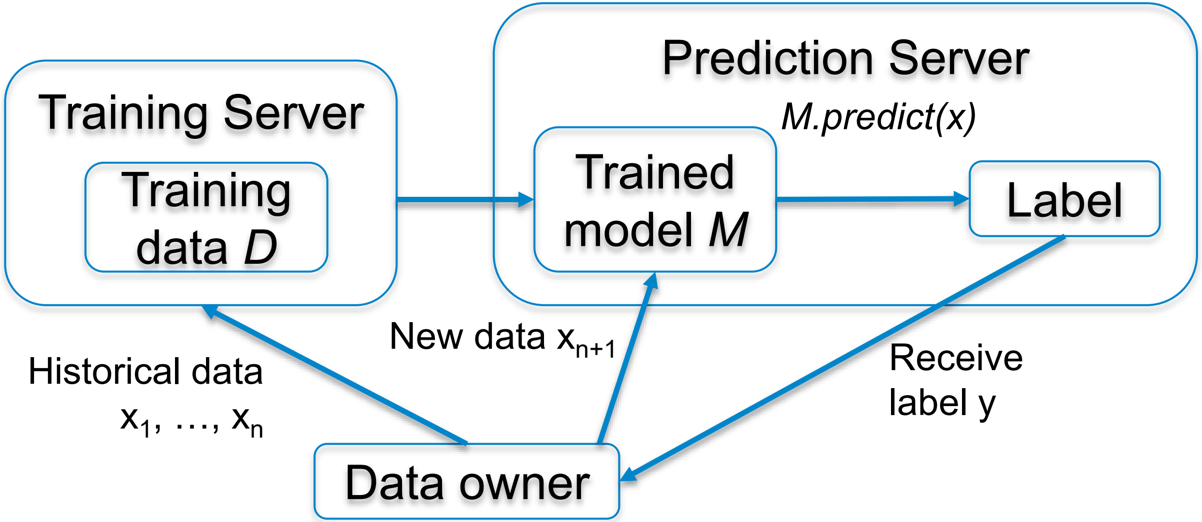
Note: FHE is the least mature of the privacy-respecting technologies discussed, but also, in principle, perhaps the most protective. FHE is hence used rarely but being aware of it as a future trend matters, especially given that cryptography is in general a technological mega-trend due to the rise of blockchain.

FHE: the two questions that matter

- **Which exact part(s) of the ML pipeline do you want to protect, and from whose perspective?**
- What type of computations are actually feasible using FHE?

Encryption of different parts of the ML pipeline

Consider a Data Owner (DO) that provides training data to a Cloud Service Provider (CSP), and then expects the CSP to train an ML model on his data, and use it to predict the labels of future data.

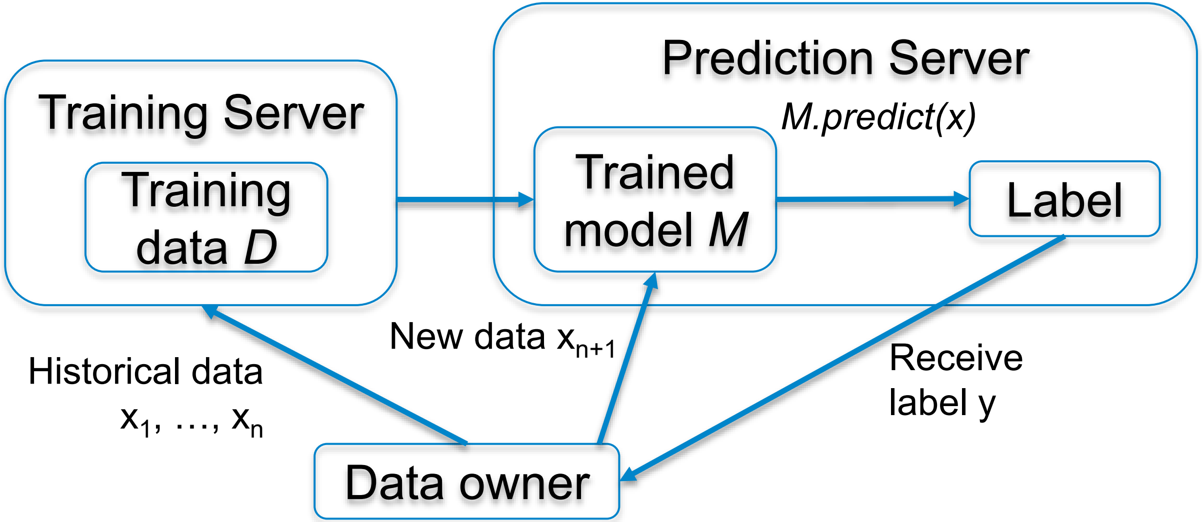


Following Graepel, Thore, Kristin Lauter, and Michael Naehrig. "ML confidential: Machine learning on encrypted data." *International conference on information security and cryptology*. Springer, Berlin, Heidelberg, 2013.

Encryption of different parts of the ML pipeline

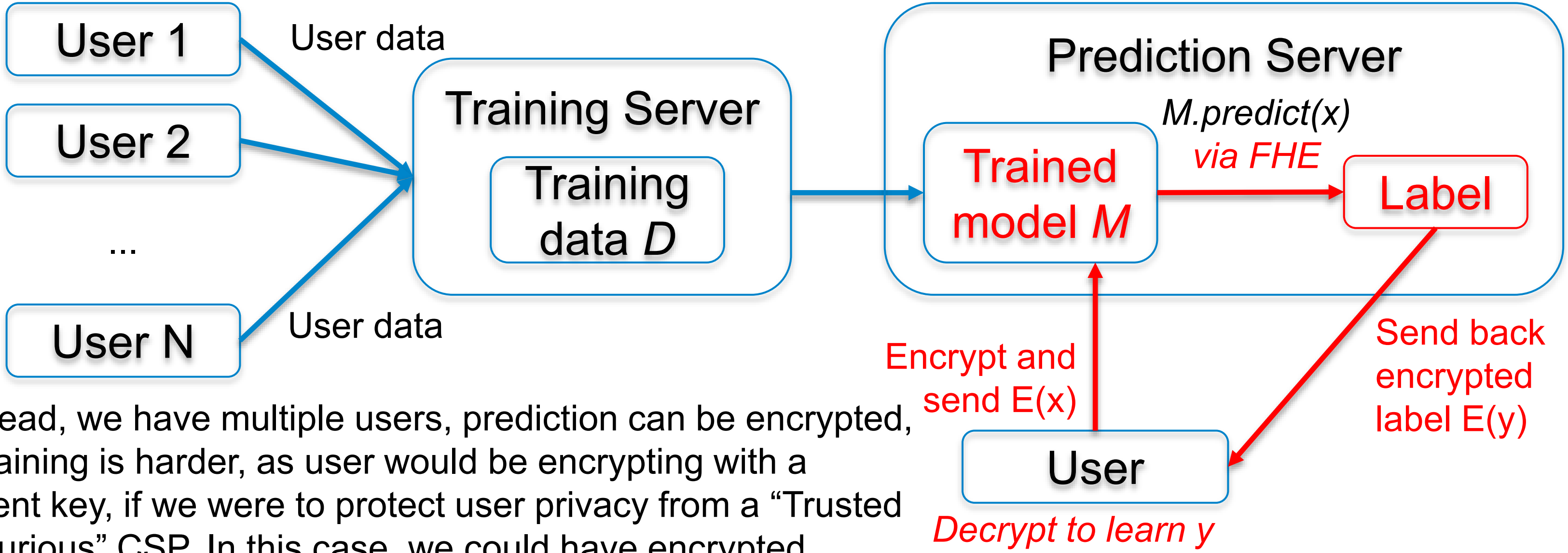
Consider a Data Owner (DO) that provides training data to a Cloud Service Provider (CSP), and then expects the CSP to train an ML model on his data, and use it to predict the labels of future data.

In this situation it is possible to do all of this on encrypted data. We will describe how.



Following Graepel, Thore, Kristin Lauter, and Michael Naehrig. "ML confidential: Machine learning on encrypted data." *International conference on information security and cryptology*. Springer, Berlin, Heidelberg, 2013.

Encryption of different parts of the ML pipeline



If instead, we have multiple users, prediction can be encrypted, but training is harder, as user would be encrypting with a different key, if we were to protect user privacy from a "Trusted but Curious" CSP. In this case, we could have encrypted prediction but plaintext training (on consented users).

An encrypted pipeline for polynomial ML for a single user

- `enc_data = he.enc(key, data)`
- `enc_f_of_data = he.eval(f, enc_data)` evaluates a polynomial `f()` on encrypted data using he.add() and he.multiply() and returns the encrypted version of `f(data)`
- Using `he.eval()` we can build `enc_model= he.ML_train(data)` to train an ML model. This will return an encrypted model (its estimated parameters will live in ciphertext space)
- Using `he.eval()`, we can also build `enc_label = he.ML_predict(new_enc_data, enc_model)` that evaluates the prediction of the ML model and returns an encrypted label
- Using `he.dec(enc_label)` the user obtains the label on their encrypted new data vector.

Note that the Cloud Service Provider now owns a valuable asset (the ML model) that she cannot, however, read. She could however run diagnostics on it using prediction queries.

Following Graepel, Thore, Kristin Lauter, and Michael Naehrig. "ML confidential: Machine learning on encrypted data." *International conference on information security and cryptology*. Springer, Berlin, Heidelberg, 2013.

FHE: the two questions that matter

- Which exact part(s) of the ML pipeline do you want to protect, and from whose perspective?
- **What type of computations are actually feasible using FHE?**

FHE: what computations does it allow?

- Addition and multiplication gives you arbitrary polynomials, e.g., $f(x) = 0.3\ x_1^2 + 0.2\ x_1x_2 - 0.3\ x_2^2$. This means you can compute a non-linear regression in an FHE manner.
- It also gives you a NOT-AND gate which in fact allows you to compute arbitrary Boolean functions:

NAND(0, 0) = 1
NAND(0, 1) = 1
NAND(1, 0) = 1
NAND(0, 1) = 0

NAND(x, y) = 1 + x*y

- NAND gates theoretically give us general computing, but in an intractable way: you would have to encrypt individual bits, not numbers, and other subtler problems arise too (e.g., how RAM works).
- *We will only focus on computation of polynomials on real numbers (in fact, integers) in this course.*

FHE: what computations does it allow?

- So how general are polynomials? Let's continue with Boolean functions as another example:

$$\begin{aligned} \mathrm{OR}(x_1, x_2, \ldots, x_n) &= \begin{cases} 0 \text{ iff } x_1 = x_2 = \cdots = x_n \\ 1, \text{ otherwise} \end{cases} \\ &= 1 - (1 - x_1)(1 - x_2) \ldots (1 - x_n) \end{aligned}$$

- Although this is an exact representation, it is a very expensive one as it is an nth order polynomial.
- In theory, there are FHE schemes that can support arbitrary degree polynomials. Practically, we can assume support up to a maximum degree D (either for reasons of computational speed, or because our scheme is a *somewhat* or *leveled* scheme which by construction is bounded degree).

FHE: what computations does it allow?

- What about a simple comparison:

$$f(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise.} \end{cases}$$

- Comparisons are unfortunately not representable as a polynomial of fixed degree D[1]. This rules out the use of simple classifiers like support vector machines, but also the use of polynomial threshold functions which would have been invaluable (e.g., $\mathrm{OR}(x_1, x_2, x_3) = x_1 + x_2 + x_3 > 0$).
- Division (so also matrix inversion) are also not representable as polynomials, nor are trigonometric functions. This rules out kernels, least squares – well, pretty much everything. How do we proceed?

1: Unless encryption happens at the level of individual bits which we have explained is out of scope here

FHE: what computations does it allow?

The solution requires careful engineering and even delegation across the parties. For example:

- Classifiers that rely on thresholding a scoring function can be implemented by letting the CSP compute the scoring function, and asking the user to compute the thresholding function, e.g.,:

```
enc_score = he.logistic_regression_predict_proba(enc_data, enc_model)
score = he.dec(key, enc_score)
label = score > 0
```

- Division (e.g., matrix inversion) can be approximated using gradient descent.
- Exponentials can be approximated by truncated Taylor expansions.
- And throughout, real numbers are represented as integers by pre-specifying a maximum precision, multiplying everything with an appropriately large number, and rounding to the nearest integer .

Conclusion

- Fully homomorphic encryption is a malleability property of a cryptosystem that allows us to perform computations on the ciphertexts without needing to decrypt the content.
- It requires careful engineering and approximation as the only practical systems currently available effectively rely on using polynomials of bounded degree and integer approximations of real numbers. As a result, privacy via encryption carries a cost in accuracy, not just in computational speed.
- It is critically important to think of encryption in the context of the full ML pipeline, as a multi-party computational system, where each party has different privacy needs and levels of trust.
- Given current interest in crypto, it is likely that FHE will increasingly form part of the ML toolbox.