Partially Encrypted Machine Learning using Functional Encryption

THÉO RYFFEL[1,2], EDOUARD DUFOUR-SANS[1], ROMAIN GAY[1,3], FRANCIS BACH[2,1], DAVID POINTCHEVAL[1,2]
[1]ENS, CNRS, PSL University, Paris, France [2]INRIA, Paris, France [3]University of California, Berkeley

Functional Encryption in Machine Learning

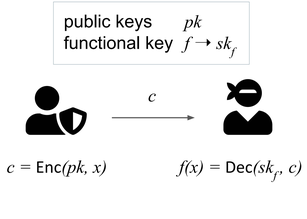
Build non-interactive encrypted machine learning as a service (MLaaS) using Functional Encryption

Use cases:

- Spam filtering: encrypted emails be classified as spam or not by the email server
- Privacy-preserving enforcement of content policies: maintaining user privacy while filtering out illegal content in messaging apps

What is Functional Encryption?

Functional Encryption allows a non-trusted party to learn the output of a specific function over encrypted data *without* interacting with the data owner.



$\underline{\text{SetUp}(1^\lambda, \mathcal{F}_{n,B_x,B_y,B_f})}$: $\mathcal{PG} := (\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, e) \leftarrow \text{GGen}(1^\lambda)$, $\vec{s}, \vec{t} \xleftarrow{\$} \mathbb{Z}_p^n$, $\text{msk} := (\vec{s}, \vec{t})$, $\text{pk} := \left(\mathcal{PG}, g_1^{\vec{s}}, g_2^{\vec{t}}\right)$ Return (pk, msk).

$\underline{\text{Enc}(\text{pk}, (\vec{x}, \vec{y}))}$:

$\gamma \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{W} \xleftarrow{\$} \text{GL}_2$, $\forall i \in [n]$, $\vec{a}_i := (\mathbf{W}^{-1})^\top \begin{pmatrix} x_i \\ \gamma s_i \end{pmatrix}$, $\vec{b}_i := \mathbf{W} \begin{pmatrix} y_i \\ -t_i \end{pmatrix}$

Return $\text{ct} := \left(g_1^\gamma, \{g_1^{\vec{a}_i}, g_2^{\vec{b}_i}\}_{i \in [n]}\right) \in \mathbb{G}_1 \times (\mathbb{G}_1^2 \times \mathbb{G}_2^2)^n$.

$\underline{\text{KeyGen}(\text{msk}, q)}$:

Return $\text{dk}_f := \left(g_2^{q(\vec{s},\vec{t})}, q\right) \in \mathbb{G}_2 \times \mathcal{F}_{n,B_x,B_y,B_q}$.

$\underline{\text{Dec}\left(\text{pk}, \text{ct} := \left(g_1^\gamma, \{g_1^{\vec{a}_i}, g_2^{\vec{b}_i}\}_{i \in [n]}\right), \text{dk}_q := \left(g_2^{q(\vec{s},\vec{t})}, q\right)\right)}$:

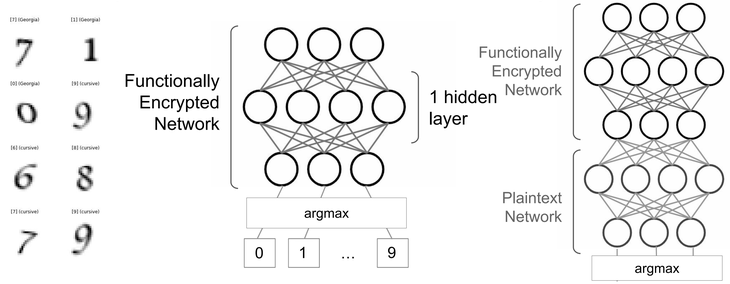
$out := e(g_1^\gamma, g_2^{q(\vec{s},\vec{t})}) \cdot \prod_{i,j \in [n]} e(g_1^{\vec{a}_i}, g_2^{\vec{b}_j})^{q_{i,j}}$

Return $\log(out) \in \mathbb{Z}$.

Figure: Our quadratic functional encryption scheme

Application to Neural Network

We propose an efficient quadratic functional encryption scheme which is equivalent to a network with one hidden layer.



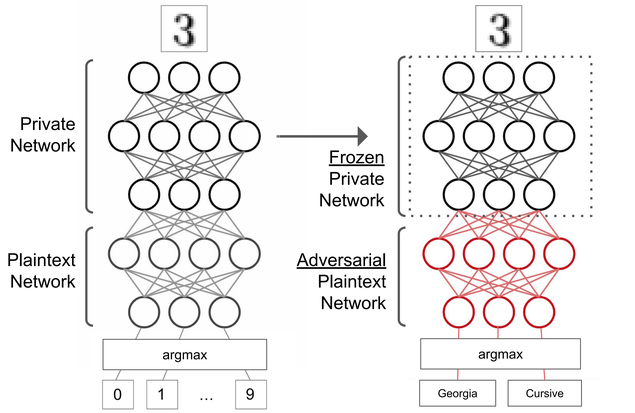
Collateral Learning, a threat to privacy

The neural network trained to perform a specific task can also be used to perform another different and highly sensitive task

An adversary can misuse the private network to perform an unexpected task and extract private information about the data. This is done below by changing the plaintext network and the target labels.

Example:

Instead of predicting the digit on the image, the adversary tries to detect the font used.



Main Task

$p \rightarrow$ 0 1 ... 9

Random baseline	**10%**
Accuracy with a CNN	**98,6%**

Collateral Task

$p \rightarrow$ Georgia Cursive

Random baseline	**50%**
Accuracy with a CNN	**93,5%**

Defeat Collateral learning with adversarial learning

We make the private model robust by simulating adversaries. We train it simultaneously on the main objective and against the collateral objective of the simulated adversary.

Semi-adversarial training protocol:

1. Optimize the main plaintext network with the private one freezed
2. Optimize the adversarial plaintext one with the private net freezed
3. Optimize the private network with the two others freezed, on the joint objective:

$$L_{joint} = L_{main} - \alpha \cdot L_{collateral}$$

where L_{main} and $L_{collateral}$ respectively are the loss of the main and adversarial networks

Robustness is further increased by simulating stronger adversaries and by reducing the output size of private network:

Main Task

Random baseline	**10%**
Accuracy with a CNN	**98,4%**

Collateral Task

Random baseline	**50%**
Accuracy - with a CNN - with best sklearn model	 **55,3%** **58,9%**

Get in touch

GitHub
github.com/LaRiffle/collateral-learning

9 detailed tutorials

arxiv.org/abs/1905.10214

References

Baltico et al. *Practical FE for quadratic functions, 2017*
Carpov et al. *Illuminating the dark, 2018*
Louppe et al. *Learning to pivot with adversarial networks, 2017*