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]École Normale Supérieure

[2]INRIA

[3]UC Berkeley

August 18, 2019

Table of Contents

Functional Encryption

Traditional PKE: all or nothing.

Functional Encryption

Traditional PKE: all or nothing.

- Have the key?
  Get the plaintext.
- Don't have the key?
  Get nothing.

Functional Encryption

Traditional PKE: all or nothing.

- Have the key?
  Get the plaintext.
- Don't have the key?
  Get nothing.

Functional Encryption: **A new paradigm**.

Functional Encryption

Traditional PKE: all or nothing.

- Have the key?
  Get the plaintext.
- Don't have the key?
  Get nothing.

Functional Encryption: **A new paradigm**.
Get a *function* of the cleartext.

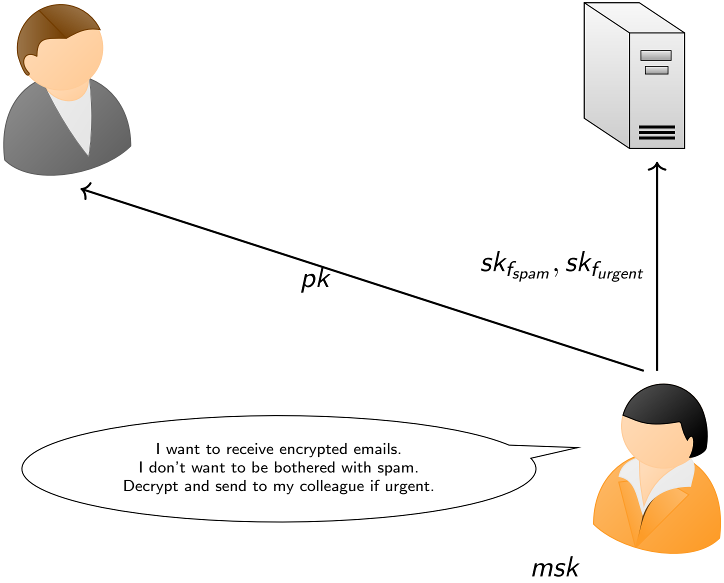
Functional Encryption

Traditional PKE: all or nothing.

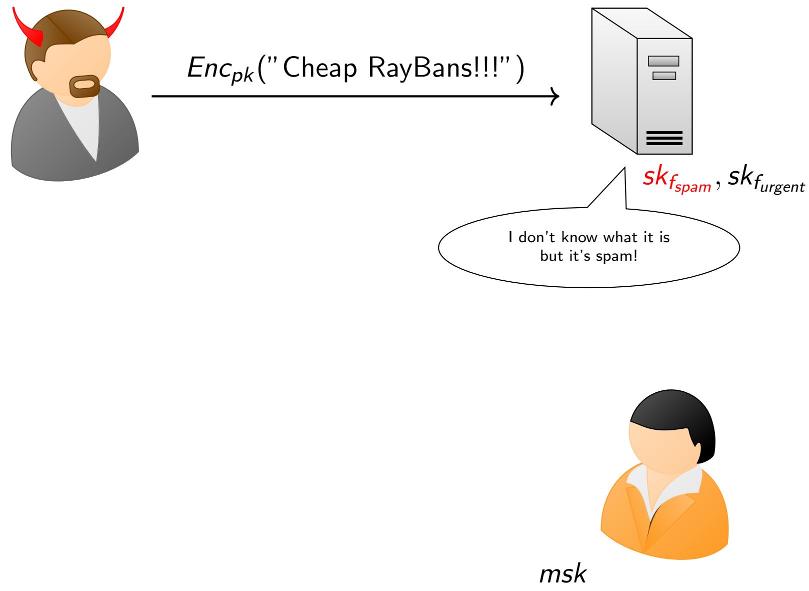
- Have the key?
  Get the plaintext.
- Don't have the key?
  Get nothing.

Functional Encryption: **A new paradigm**.
Get a *function* of the cleartext.
Function depends on the key.

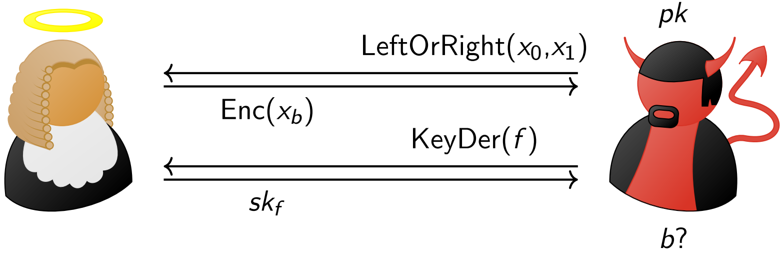
FE example


$sk_{f_{spam}}, sk_{f_{urgent}}$
pk
I want to receive encrypted emails.
I don't want to be bothered with spam.
Decrypt and send to my colleague if urgent.
msk

FE example


$Enc_{pk}(\text{"Cheap RayBans!!!"})$
pk
$sk_{f_{spam}}, sk_{f_{urgent}}$
I don't know what it is but it's spam!
msk

Security definitions



Security definitions

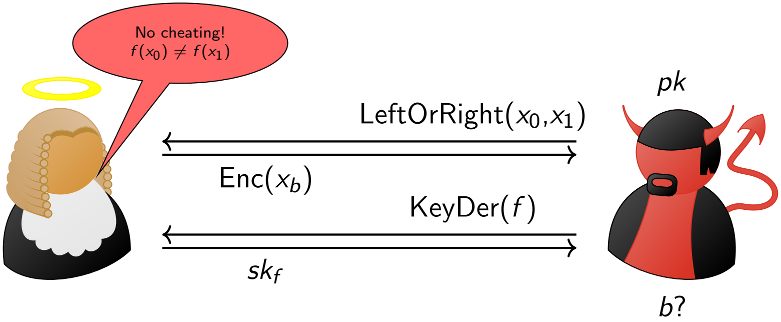


Table of Contents

Our contributions

- New Quadratic FE scheme;
- Python Implementation;
- Methodology for Thinking About Privacy in FE-ML;
- New Dataset;
- Collateral Learning Framework for Training Models in FE-ML.

Table of Contents

A New FE Scheme for Quadratic Forms

- Key $sk_{\mathbf{Q}}$ gets you $\vec{x}^T \mathbf{Q} \vec{x}$ from $Enc(\vec{x})$;
- Decryption $1.5\times$ faster than State-of-the-Art;
- Uses pairings. Secure in Generic Group Model;

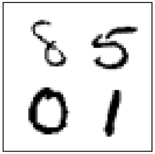
A New FE Scheme for Quadratic Forms

- Key $sk_{\mathbf{Q}}$ gets you $\vec{x}^T \mathbf{Q} \vec{x}$ from $Enc(\vec{x})$;
- Decryption $1.5\times$ faster than State-of-the-Art;
- Uses pairings. Secure in Generic Group Model;
- All group-based computational FE schemes require a discrete logarithm;
- Must ensure output has reasonably small entropy;

A New FE Scheme for Quadratic Forms

- Key $sk_{\mathbf{Q}}$ gets you $\vec{x}^T \mathbf{Q} \vec{x}$ from $Enc(\vec{x})$;
- Decryption $1.5\times$ faster than State-of-the-Art;
- Uses pairings. Secure in Generic Group Model;
- All group-based computational FE schemes require a discrete logarithm;
- Must ensure output has reasonably small entropy;
- All DLOGs are in base g_T!
- We precompute tweaked Giant step of BSGS and store for reuse.

A Simple Model



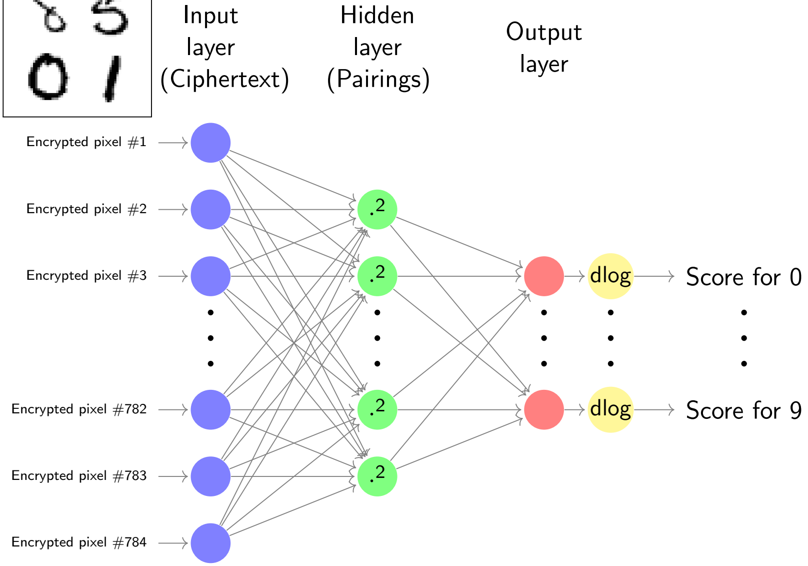


Table of Contents

Leakage

Ciphertexts are for vectors $\vec{x} \in [0, 255]^{784}$.
A key for $\mathbf{Q}$ lets you compute one scalar $\vec{x}^T \mathbf{Q} \vec{x}$.

Leakage

Ciphertexts are for vectors $\vec{x} \in [0, 255]^{784}$.
A key for $\mathbf{Q}$ lets you compute one scalar $\vec{x}^T \mathbf{Q} \vec{x}$.
More keys give you more scalars.

Leakage

Ciphertexts are for vectors $\vec{x} \in [0, 255]^{784}$.
A key for $\mathbf{Q}$ lets you compute one scalar $\vec{x}^T \mathbf{Q} \vec{x}$.
More keys give you more scalars.
But your notion of privacy depends on the distributions on the $\vec{x}$'s.

Leakage

Ciphertexts are for vectors $\vec{x} \in [0, 255]^{784}$.
A key for $\mathbf{Q}$ lets you compute one scalar $\vec{x}^T \mathbf{Q} \vec{x}$.
More keys give you more scalars.
But your notion of privacy depends on the distributions on the $\vec{x}$'s.
10 scalars actually give a lot of information: [CFLS18] mount good recovery attacks.

Defining Security for FE-ML

Security definition of FE isn't very helpful for deciding how many keys you can give out.

Defining Security for FE-ML

Security definition of FE isn't very helpful for deciding how many keys you can give out.
What information are we trying to protect?

Defining Security for FE-ML

Security definition of FE isn't very helpful for deciding how many keys you can give out.
What information are we trying to protect?
Is a decent reconstruction of a MNIST image bad for privacy? Is it ok? Which details matter?

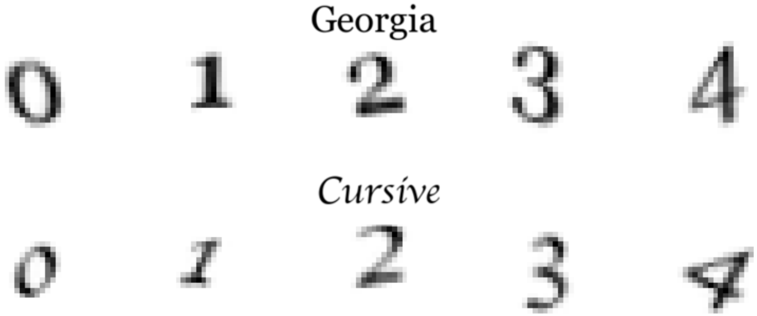
Defining Security for FE-ML

Security definition of FE isn't very helpful for deciding how many keys you can give out.
What information are we trying to protect?
Is a decent reconstruction of a MNIST image bad for privacy? Is it ok? Which details matter?
We need to capture real-world concerns on real-world data distributions.

Defining Security for FE-ML

Security definition of FE isn't very helpful for deciding how many keys you can give out.
What information are we trying to protect?
Is a decent reconstruction of a MNIST image bad for privacy? Is it ok? Which details matter?
We need to capture real-world concerns on real-world data distributions.
We can draw inspiration from the cryptographic notion of indistinguishibility.

Defining Security for FE-ML



Collateral Learning

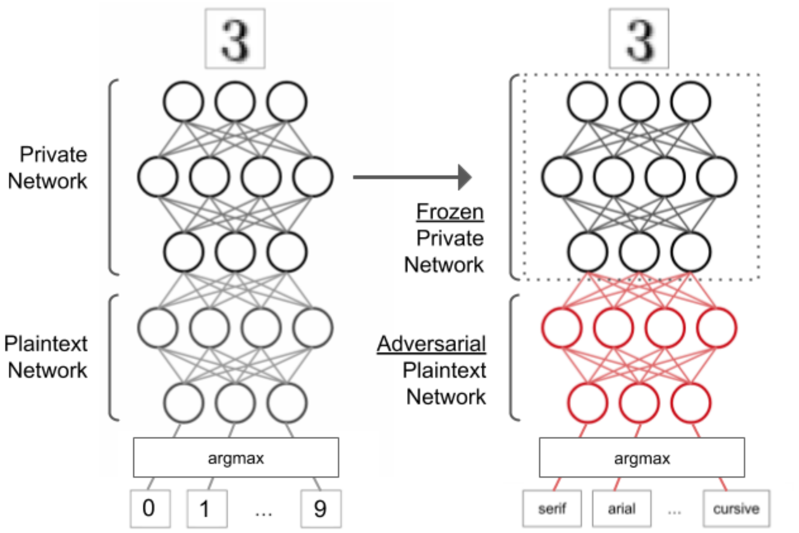


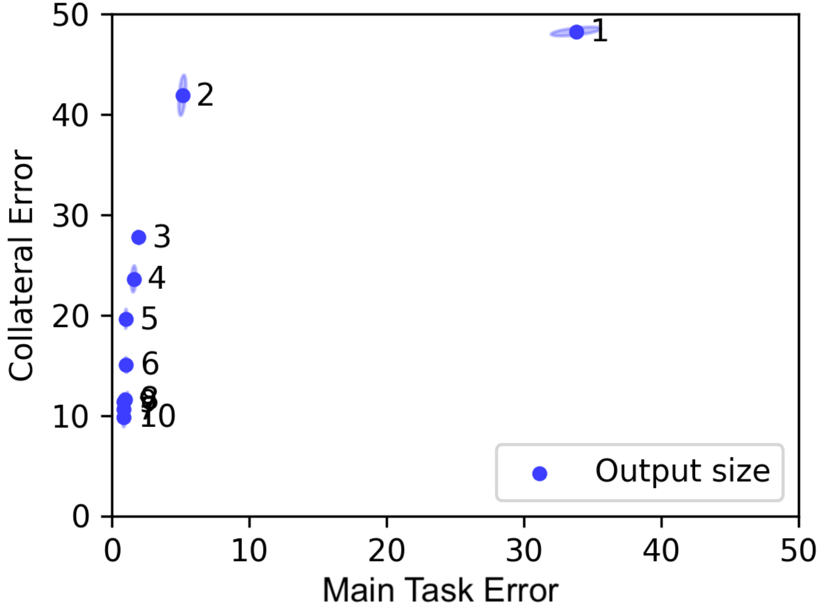
Table of Contents

Implementation

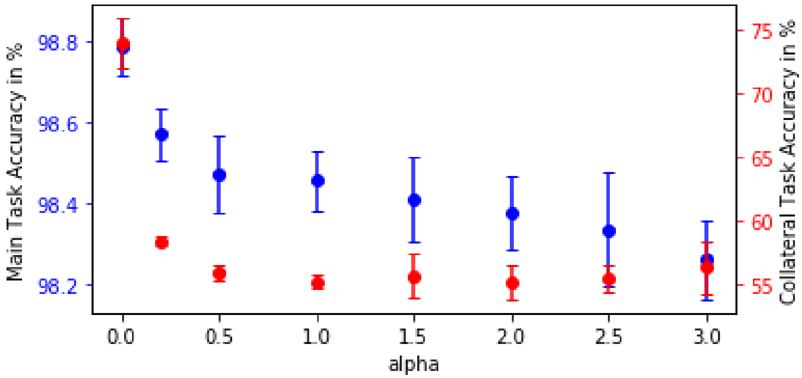
We provide a Python implementation using Charm with PBC.
We use a database for precomputed discrete logarithms.

Functional key generation	0.094s
Encryption time	12.1s
Evaluation time	2.97s
Discrete logarithms time	0.024s

Results: Influence of Output Size



Results: Influence of Adversarial Parameter



Open problems

- Bigger images.

Open problems

- Bigger images.
- Richer FE.

Open problems

- Bigger images.
- Richer FE.
- Trusting models.

Recap: Our contributions

- New Quadratic FE scheme;
- Python Implementation;
- Methodology for Thinking About Privacy in FE-ML;
- New Dataset;
- Collateral Learning Framework for Training Models in FE-ML.