PAGE NO.
DATE : / /

2/08/20

Partially Encrypted Machine Learning Using Functional Encryption

Tags:- Functional Encryption, quad activation

* What they present →
  - efficient computation of quad functions
  - learn a specific functional eval. of some encryted data
  - Partially encrypt N.N. with quad. activation fns.; info leaks based on indistinguishability of data items of same bb label.
  - adversarily optimizes the network against an adversary trying to identify these features.
  - little reduction in perf., but significant improvement in privacy.

* for a function - f, a functional decryption key can be generated such that given any ciphertext with underlying plain-text x, a user can use this key to obtain f(x) without learning about x / any other info other than f(x)
  * Its b/w regular encryption and FHE
    - regular encryption → data can be revealed directly
    - FHE → can only manipulate, can't disclose anything much

* Use cases → spam filtering, content filtering basically.

* 1st layers of a polynomial network can be run on encrypted I/ps using this scheme!!

PAGE NO.
DATE : / /

* Adversarial Training Technique :- process the ist layers so that to improve their privacy, so that their output which in plain text cannot be used by adversaries at test time.
  * this decrypted output ~~in~~ isn't directly the classification result but as an ~~the~~ intermediate layer.

① Background Knowledge :-

(1.1) Quad / Polynomial Neural Networks :-

* use only linear elements like → FC ~~Linear~~
  → Convolution
  → avg pooling
* Non-linear activations are replaced by polynomial approximations (when simply not a square function)
* However, the argmax / thresholding fn. present at the end of a classifier cannot be conviniently handled so, they run polynomial networks on encrypted I/ps but take the argmax over the decrypted network.

(1.2) Functional Encryption :-

* public key - pk → used to encrypt the data
  secret key - sk → " " derive a functional decryption key dk_f
  ie. given a ciphertext of x, can decrypt f(x)

perfect correctness when,

$\forall x \in X$ and $f \in F$

$$Pr[Dec(dk_f, ct) = f(x)] = 1$$

where $dk_f \leftarrow KeyGen(msk, f)$

and $ct \leftarrow Enc(pk, x)$

(1.3) Indistinguishability / security :-

* make sure we cannot learn anything more than f.
* how sensitive f(x) is to x.
* indistinguishability ⇒ given an encrypted and non-encrypted o/p an adversary must not detect which I/P

PAGE NO.
DATE : / /

* q - quadratic FE scheme aims at predicting y_{pub} and an adversary to infer y_{priv}
So, adversary provides 2 inputs (x_0, x_1) with same y_{pub} but different y_{priv}. Then try to distinguish which one was selected by the challenger given $q(x_b)$ $\hookrightarrow b \in \{0,1\}$

* Separation :- y_{priv} to be Independent from $q(x)$ given the true label y_{pub}

(2) Context for Private Inference :-

(2.1) Classifying in Two Directions:

* Data-sets having $(x_i)_{i=1,..n}$ which have public labels y_{pub} but also private ones y_{priv}
eg: y_{pub} → spam flag
y_{priv} → marketing info highlighting areas of interest of the recipient.

* Dataset prepared → fonts dataset of numbers
digit y_{pub}
font y_{priv}

* 2 Tasks Here

maximize ↑ * predict y_{pub} using a partially encrypted polynomial N.N. with FE

minimize ↓ * adversary leverage the o/p of the FE network at test time to predict y_{priv}

(2.2) Equivalence with a Quad Functional Encryption Scheme

(2.2.1) FE for Quad Polynomials

$$q(\vec{x}, \vec{y}) = \sum_{i,j \in [n]} q_{ij}\, x_i\, y_j$$

$\{q_{i,j} \in [-B_q, B_q]\}_{i,j \in [n]}$

Set of bounded coeff

PAGE NO.
DATE : / /

NOTE:-

Bilinear groups forming a bilinear map $e: G_1 \times G_2 \to G_T$ (pairing)

and satisfies $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$

- g_1, g_2 → generators of G_1, G_2
- and $g_T := e(g_1, g_2)$ → generator of G_T

* A pair of vectors $(\vec{s}, \vec{t})$ are selected and made the mask, public key is $(g_1^{\vec{s}}, g_2^{\vec{t}})$
  * Encryption consists of making $g_1^{\vec{x}}, g_2^{\vec{y}}$ with $g_1^{\vec{s}}, g_2^{\vec{t}}$ allowing any user to compute $g_T^{q(\vec{x},\vec{y}) - q(\vec{s},\vec{t})}$
  * Decryption key for a specific q is $g_T^{q(\vec{s},\vec{t})}$ allowing to get $g_T^{q(\vec{x},\vec{y})}$
* After discrete logarithms, we have access to $g_T^{q(\vec{x},\vec{y})}$