The 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019)
Dallas, Texas, US

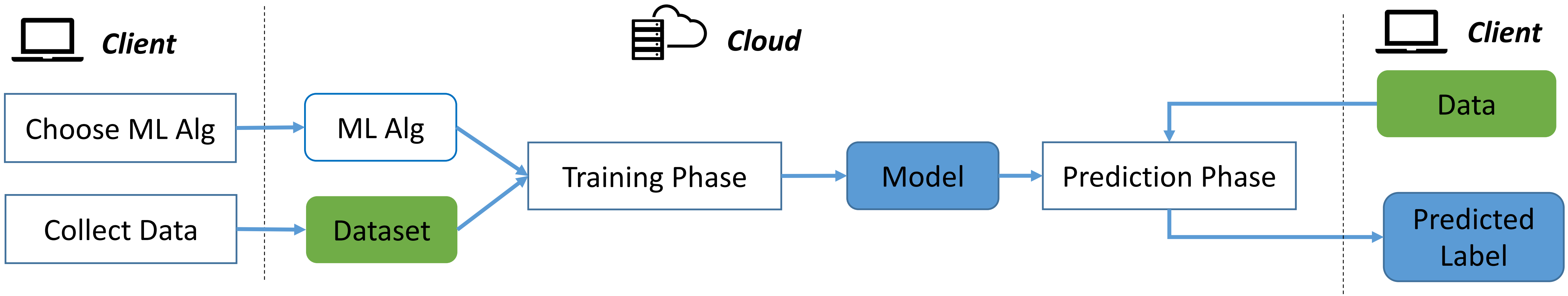
CryptoNN: Training Neural Networks over Encrypted Data

Runhua Xu, James Joshi and Chao Li

runhua.xu@pitt.edu

Background

Cloud-based ML Service



Special Scenario, e.g., Small Clinics - Computer Aided Diagnostic Application

Challenges

Limited IT infrastructure and AI resources/experts

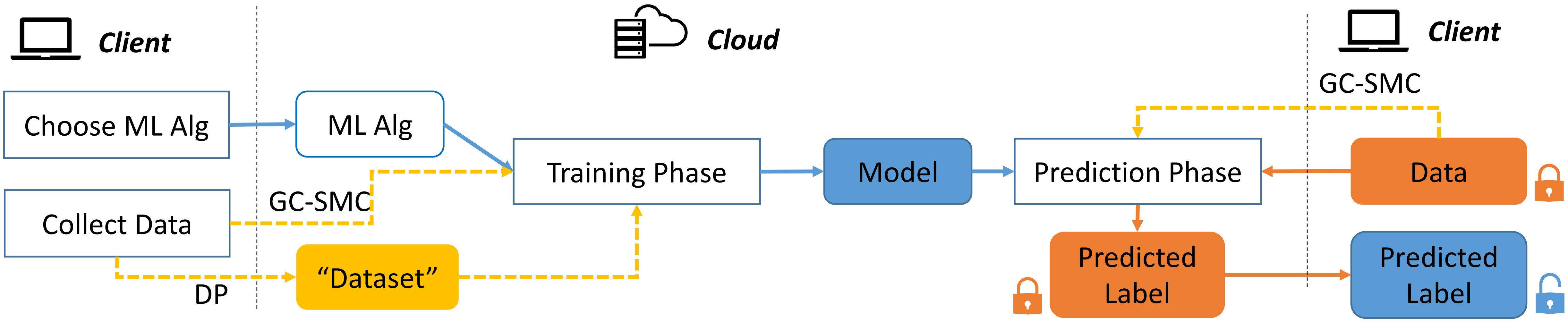
v.s.

Privacy-sensitive data – e.g., patients' electronic healthcare records

How to train a ML model without leaking privacy-sensitive data using cloud-based ML service ?

Background

How existing privacy-preserving ML approaches work in cloud-based service



Privacy-Preserving Approaches

- Noise Addition
    - Differential Privacy, e.g., deep learning with differential privacy.
- Secure Multiparty Computation (SMC)
    - "non-crypto" based approach – garbled circuit (GC) + oblivious transfer (OT), e.g., DeepSecure, etc.
    - "crypto" based approach – homomorphic encryption (HE), e.g., CryptoNets, etc.

Background

Adoption of Privacy-Preserving Approaches in ML Cloud : Trade-off Issue

- ❖ Noise Addition
  - Differential Privacy

  trade-off : privacy v.s. utility

- ❖ Secure Multiparty Computation (SMC)

  trade-off : computation v.s. transmission

  - "non-crypto" based approach – GC + OT

    require large transmission volume

  - "crypto" based approach – HE

    require higher computation time
    --only support prediction phase

Comparison of Privacy-preserving ML Approaches

COMPARISON OF PRIVACY-PRESERVING APPROACHES IN MACHINE LEARNING MODELS

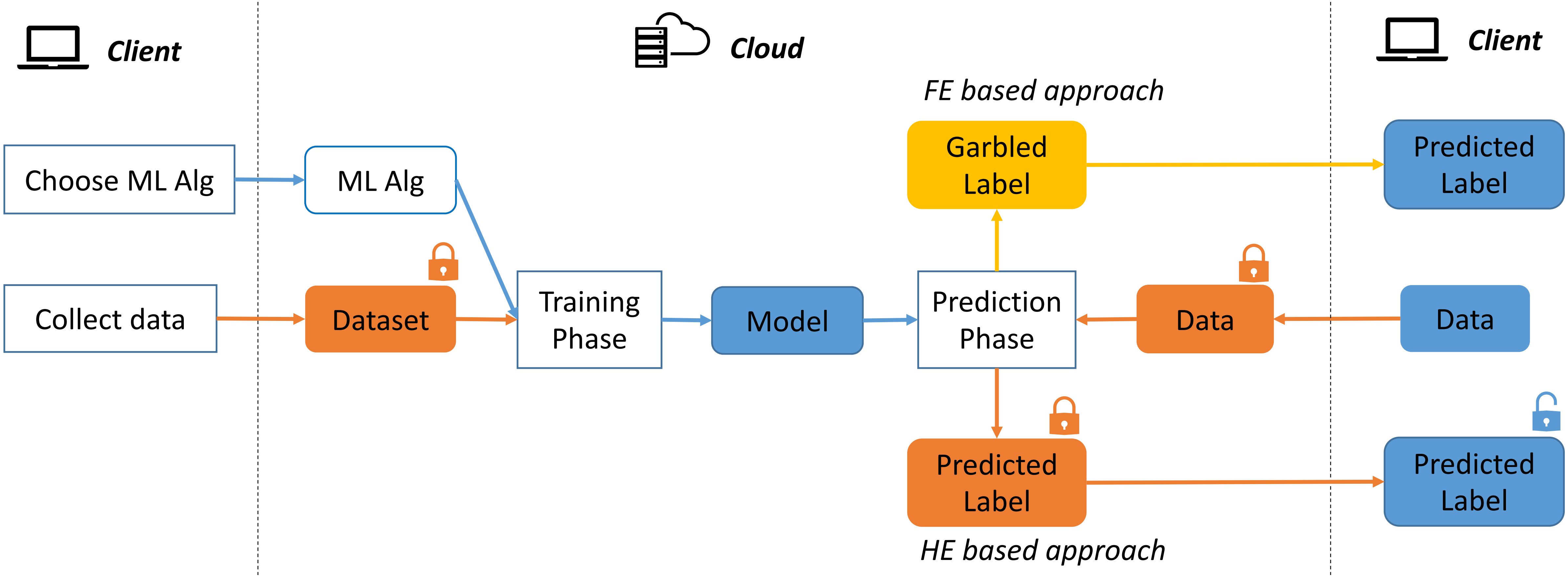
Proposed Work	Training	Prediction	Privacy ▷	ML Model	Approach
Privacy-Preserving Deep Learning (CCS) [7]	●	○	○	Deep Learning	Distributed*
Deep Learning with differential privacy (CCS) [8]	●	○	○	Deep Learning	Differential Privacy◇
CryptoML [4]	●	●	◑	Matrix-based ML	Delegation‡
SecureML [6]	●	●	◑	General	Secure Protocol (SMC)
DeepSecure [5]	●	●	◑	Deep Learning	Secure Protocol (Garbled Circuits)
CryptoNets [3], [9], [10], [11], [12], [13], [14], [15]	○	●	●	Covers All	Homomorphic Encryption (HE)
ML classication over encrypted data (NDSS) [2]	○	●	●	Limited ML†	HE + Secure Protocol
CryptoNN (our work)	●	●	●	Neural Networks	Functional Encryption

▷ This column indicates the privacy strength/guarantee such as mild approach ○ (e.g. differential privacy) and strong guarantee ● (e.g. crypto system).
† It only supports Hyperplane Decision, Nave Bayes, and Decision Trees models.
‡ The data owner trains the model by itself and outsources partial computation in a privacy-preserving setting.
* The model is trained in a distributed manner where each data owner trains a partial model on their private data.
◇ It applies differential privacy method on the training data.

CryptoNN in Cloud-based ML Service

How CryptoNN works in cloud-based ML service

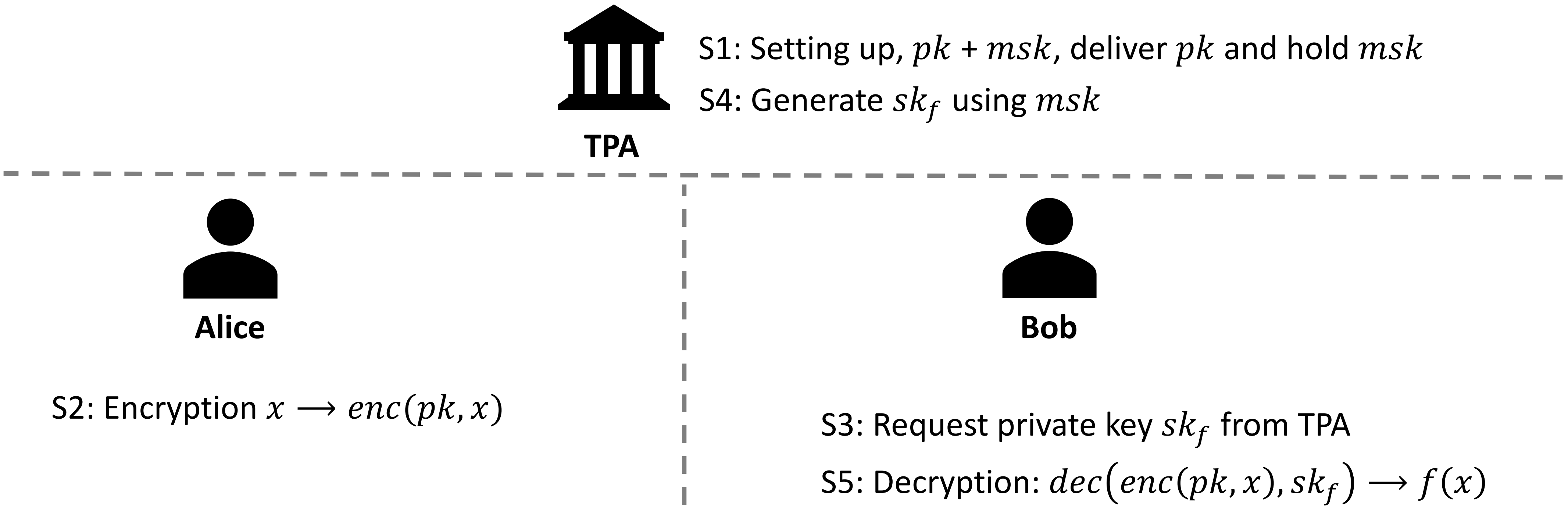
Cloud/Server based ML (as a Service) -- Clients



Functional Encryption

In traditional encryptions scheme, *decryption algorithm reveals all or nothing*

In FE, for a function $f(\cdot)$, the decryption key sk_f only <u>reveals partial information</u>, i.e., $f(x)$ instead of x.

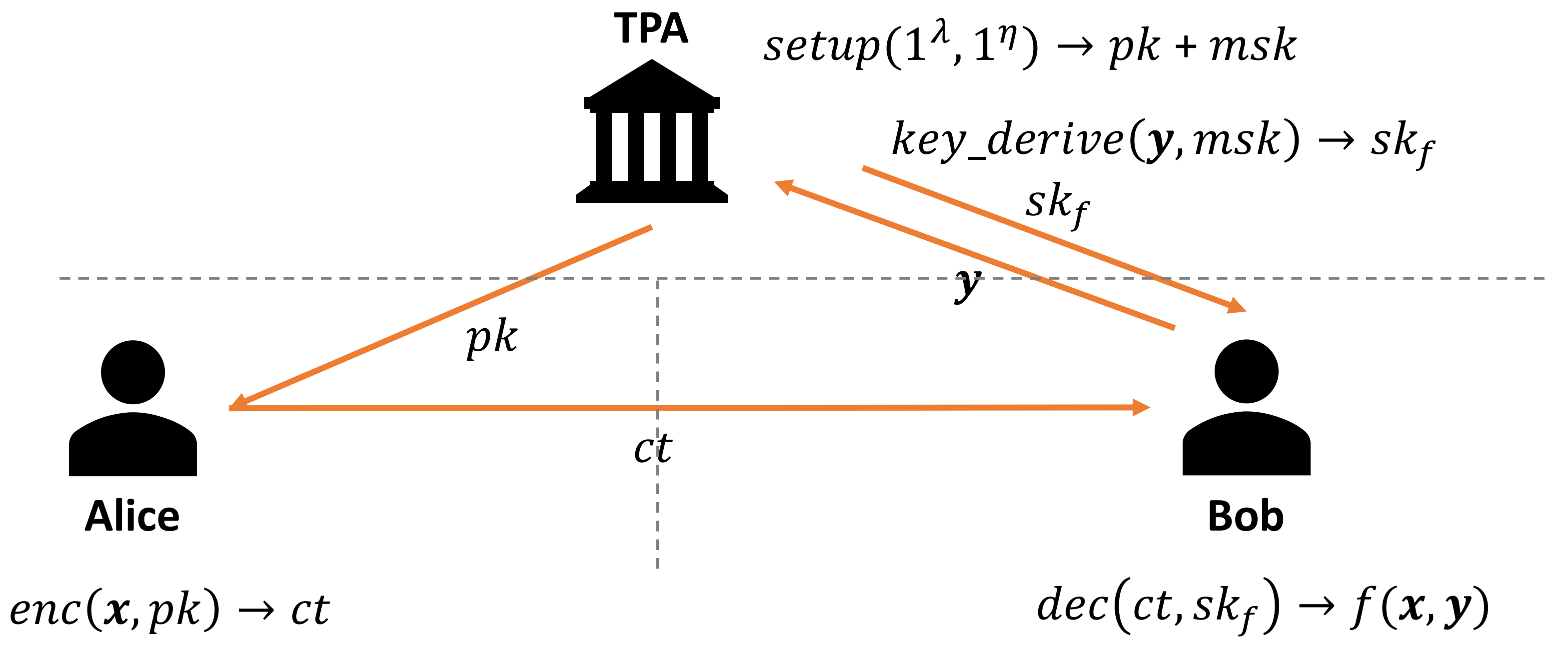


Functional Encryption -- Inner-Product

$$f(\boldsymbol{x},\boldsymbol{y}) =< \boldsymbol{x},\boldsymbol{y} > = \sum_{i=1}^{n}(x_i \cdot y_i)$$

Alice has $\boldsymbol{x}$, and Box has $\boldsymbol{y}$.

Goal:
Let Bob know the result of $f(\boldsymbol{x},\boldsymbol{y})$,
but is not able to learn $\boldsymbol{x}$.



Abdalla, Michel, et al. "Simple functional encryption schemes for inner products." IACR International Workshop on Public Key Cryptography (PKC 2015). Springer, Berlin, Heidelberg, 2015.

Secure Matrix Computation

Two Parties $\boldsymbol{X}_{\boldsymbol{l\times n}}, \boldsymbol{Y}_{\boldsymbol{n\times m}},$ s.t. $\mathrm{n} > m$

Alice

$$\begin{bmatrix} \boxed{x_{11} & \cdots & x_{1n}} \\ \vdots & \ddots & \vdots \\ x_{l1} & \cdots & x_{ln} \end{bmatrix}$$

$\boldsymbol{X}_{\boldsymbol{l\times n}}$

$ct_1 \leftarrow enc(\boldsymbol{x}_1)$

$ct_l \leftarrow enc(\boldsymbol{x}_l)$

$enc(\boldsymbol{X}) = (ct_1, \dots, ct_l)$

Bob

$$\begin{bmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nm} \end{bmatrix}$$

sk_{f_1} sk_{f_m}

$\boldsymbol{Y}_{\boldsymbol{n\times m}}$

$sk_{f,\boldsymbol{Y}} = (sk_{f_1}, \dots, sk_{f_m})$

Decryption

Computation

$dec(ct_1, sk_{f_1})$

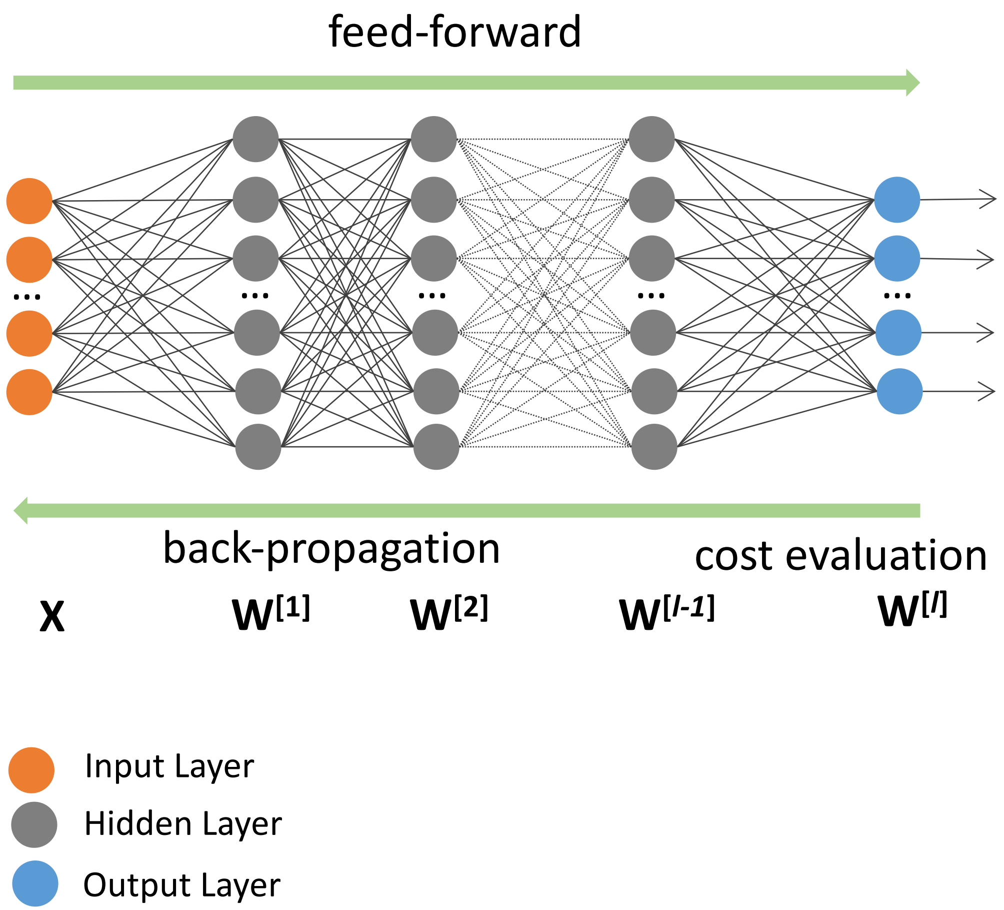
$dec(ct_1, sk_{f_m})$

$$\begin{bmatrix} z_{11} & \cdots & z_{1m} \\ \vdots & \ddots & \vdots \\ z_{l1} & \cdots & \boxed{z_{lm}} \end{bmatrix}$$

$dec(ct_l, sk_{f_m})$

$\boldsymbol{XY}_{\boldsymbol{l\times m}}$

Neural Networks - Gradient Descent



$\boldsymbol{A}^{[1]} = g(\boldsymbol{Z}^{[1]}),\ \boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$

$\boldsymbol{A}^{[2]} = g(\boldsymbol{Z}^{[2]}),\ \boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{A}^{[1]} + \boldsymbol{b}^{[2]}$

… ….

$\boldsymbol{A}^{[l-1]} = g(\boldsymbol{Z}^{[l-1]}),\ \boldsymbol{Z}^{[l-1]} = \boldsymbol{W}^{[l-1]}\boldsymbol{A}^{[l-2]} + \boldsymbol{b}^{[l-1]}$

$\boldsymbol{A}^{[l]} = g(\boldsymbol{Z}^{[l]}),\ \boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

$\widehat{\boldsymbol{Y}} = \boldsymbol{A}^{[l]}$

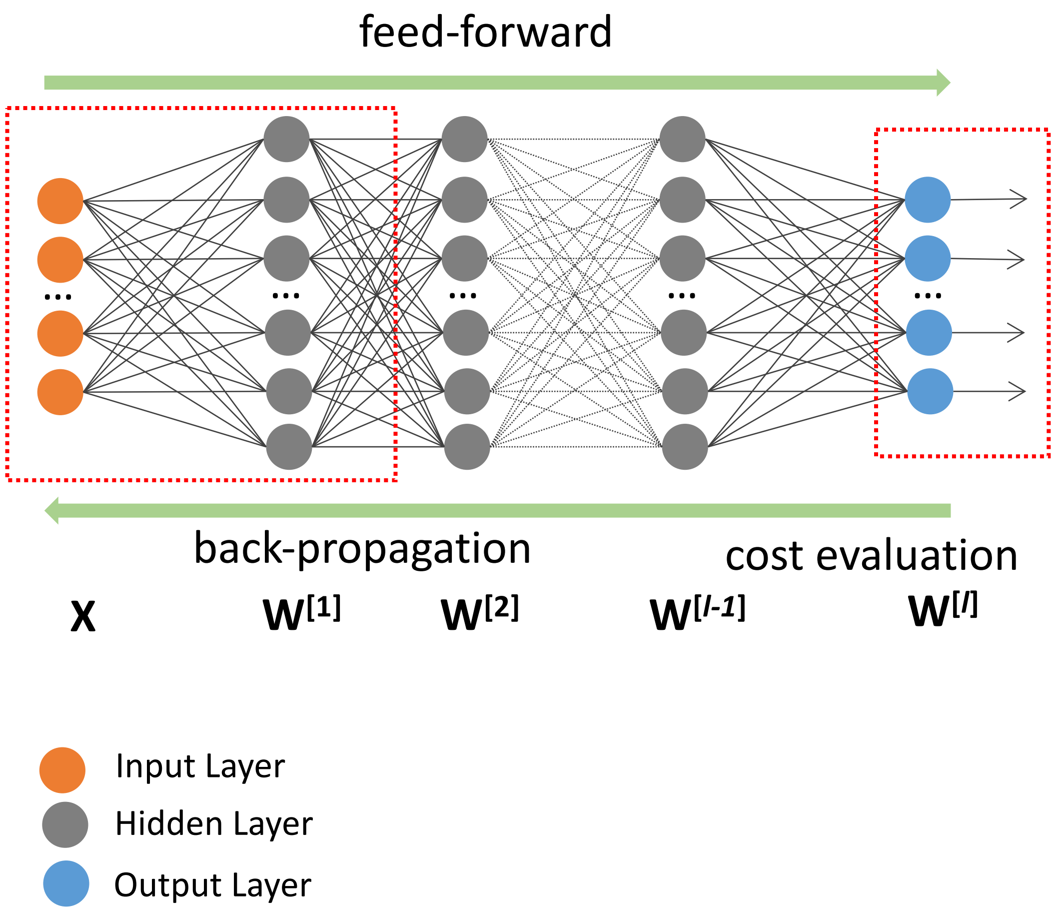
$$E = \frac{1}{n}\sum_{i}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 \qquad g(z) = \frac{1}{1+e^{-z}}$$

$$\boldsymbol{W}^{[l]} = \boldsymbol{W}^{[l]} - \alpha\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}^{[l]}}$$

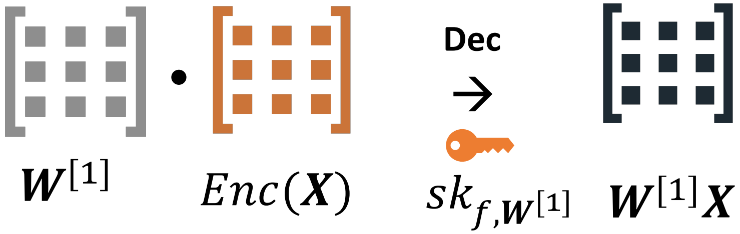
$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}^{[l]}} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{A}^{[l]}}\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{Z}^{[l]}}\frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}^{[l]}}$$

$$\frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}^{[l]}} = \boldsymbol{A}^{[l-1]},\ \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{Z}^{[l]}} = \boldsymbol{A}^{[l]}\left(1 - \boldsymbol{A}^{[l]}\right),\ \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{Z}^{[l]}} = \boldsymbol{A}^{[l]} - \boldsymbol{Y}$$

Neural Networks meet Functional Encryption



$$\boldsymbol{A}^{[1]} = g(\boldsymbol{Z}^{[1]}),\ \boldsymbol{Z}^{[1]} = \boxed{\boldsymbol{W}^{[1]}\boldsymbol{X}} + \boldsymbol{b}^{[1]}$$



Secure Matrix Computation

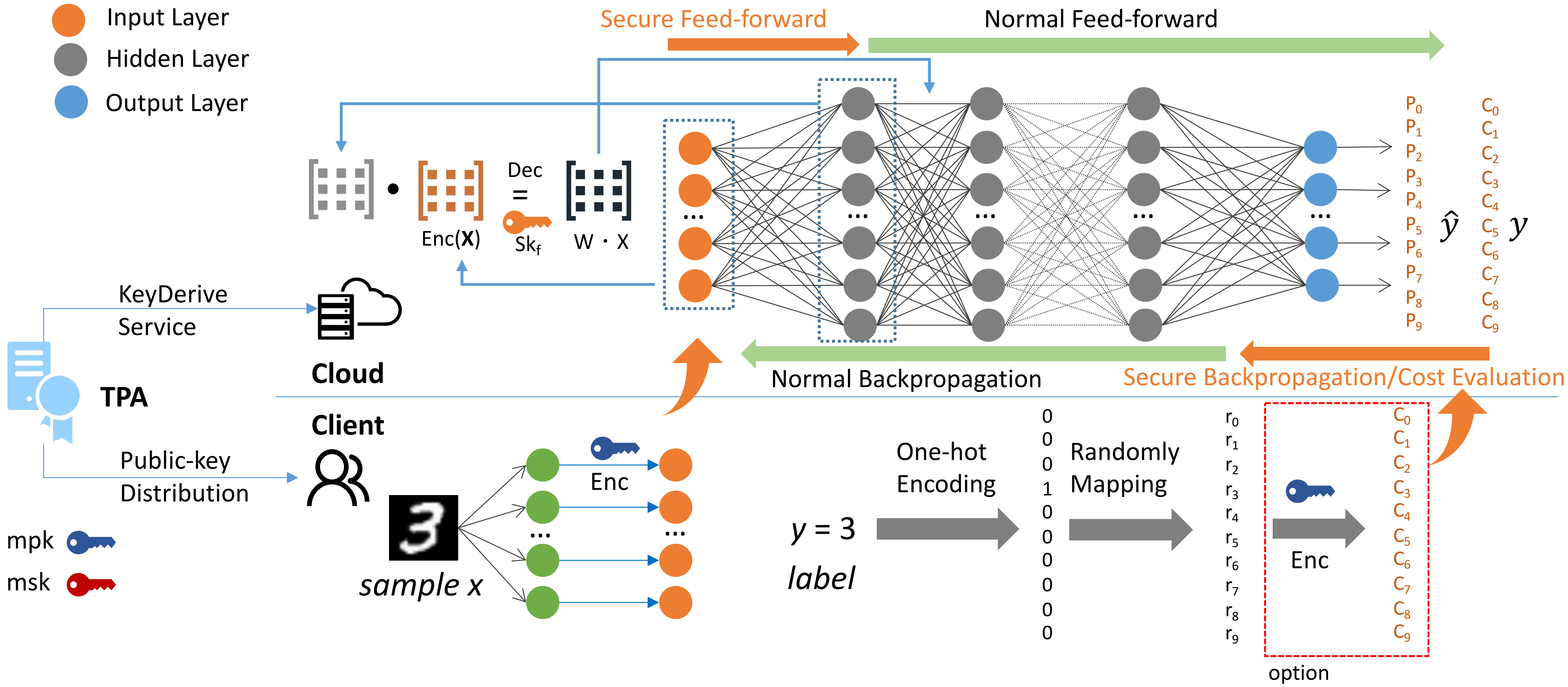
$$\boldsymbol{W}^{[l]} = \boldsymbol{W}^{[l]} - \alpha \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}^{[l]}}$$

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}^{[l]}} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{A}^{[l]}} \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{Z}^{[l]}} \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}^{[l]}}$$

$$\frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}^{[l]}} = \boldsymbol{A}^{[l-1]},\ \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{Z}^{[l]}} = \boldsymbol{A}^{[l]}(1 - \boldsymbol{A}^{[l]}),\ \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{Z}^{[l]}} = \boxed{\boldsymbol{A}^{[l]} - \boldsymbol{Y}}$$

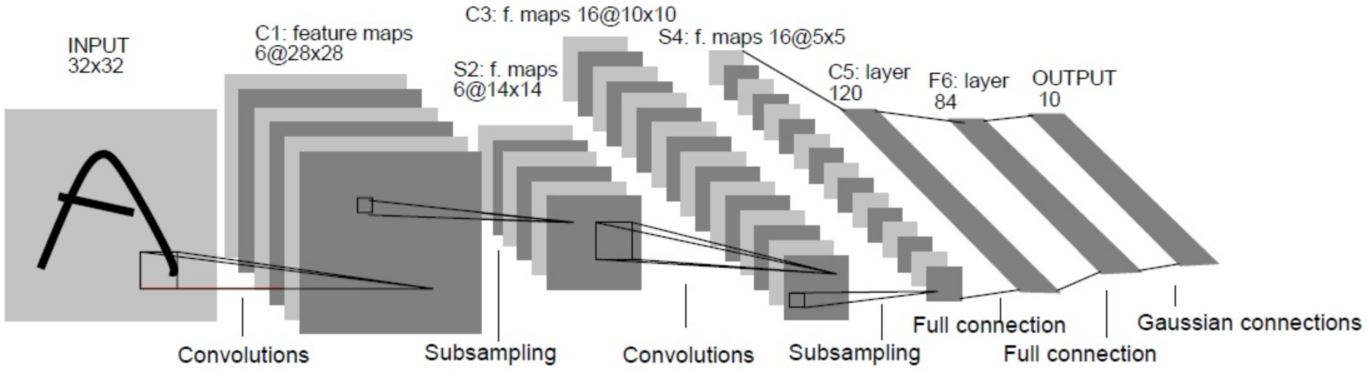
FE for Basic Operations or Garbled Labels

CryptoNN – Framework Overview



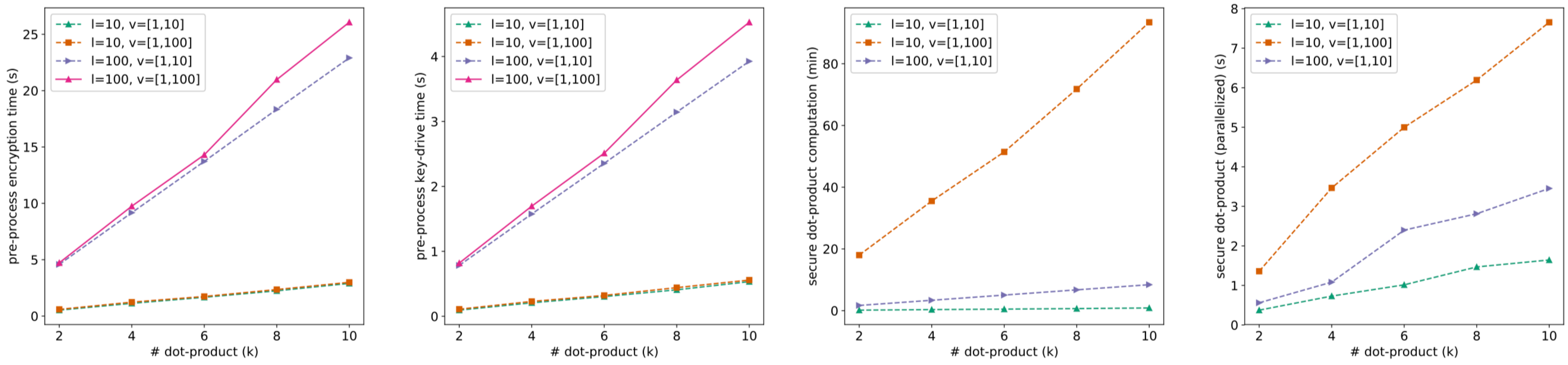
Experimental Evaluation

- Prototype Implementation
  - A scratch implementation of LeNet-5 in Python
  - *FE scheme implementation*
    - Charm-crypto (Python) – underlying numerical calculations rely on GMP library (C)
- Test platform
  - *Intel Core i7/16GB/macOS*



Experimental Evaluation

Time cost of dot-product in secure matrix computation



(a) pre-processing for encryption (b) pre-processing for function key (c) secure dot-product computation (d) secure dot-product (parallelized)

x-axis: the element size

y-axis: the computing time

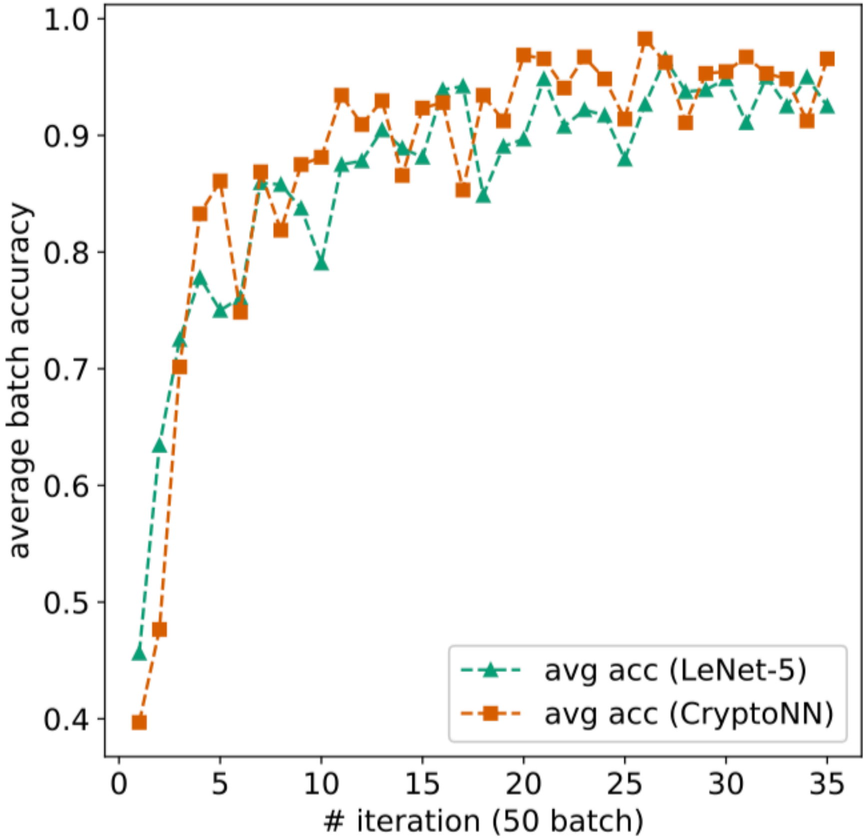
e.g., one intuitive result

$$\left.\begin{aligned} x &= (x_1, x_2, \dots, x_{10}) \\ y &= (y_1, y_2, \dots, y_{10}) \end{aligned}\right\} \; x, y : 100 \quad \rightarrow \text{ 7-8 seconds (parallelized)}$$

$$x_i, y_i \in [1,100]$$

$$X^{100\times10} \cdot Y^{10\times100} = XY^{100\times100}$$

Experimental Evaluation



model	epoch 1 (acc)	epoch 2 (acc)	training time
LeNet-5	93.04%	95.48%	4h
CryptoCNN	93.12%	95.49%	57h

LetNet-5 Neural Networks

MNIST dataset

60000 training / 10000 test

Hyper Parameters Setting

Float Point Precision Setting: 2

-- the # of bits used after the decimal point of a floating point number

-- encoding floating point number → integer number

Bath Size: 64

Learning Rate: 5e-4

Comparing to baseline:

-- achieving similar average batch accuracy

-- costing about 14 times training time

Note this is result of submitted version.

In our follow-up work, we have an efficient implementation of decryption : $X^{1\times25} \cdot Y^{25\times1}$ from 40s → 0.2ms

Summary

- CryptoNN framework
  - *Secure multiparty computation based on FE*
  - *CryptoNN framework*
  - *Concrete instance, CryptoCNN*
  - *Evaluation Results*
- Future work
  - *More efficient approaches*
  - *Prevent intermediate model inference attack*
  - *Other NN architecture*

Thanks

Q & A