Hackathon-ND Report

Song Bian[1], Dewen Zeng[2], Xinrong Hu[2], Qing Lu[2], and Weiwen Jiang[2]

[1]Kyoto University
[2]University of Notre Dame

1 Motivation

Recently, deep neural networks (DNNs) have been widely used in medical applications (e.g., medical image computing and disease diagnosis) because of their high efficiency and extremely prediction accuracy. However, the strict security regulations placed on medical records hinder the use of big data in machine learning (ML). Access controls and client-side encryption are mandated for the distribution of patient records over public network. In order to deploy DNN models to the real-world for medical image analysis, the key problem is how to handle the data transfer and computation securely and efficiently. This leads to our key motivation, we propose a secure protocol for the UNET architecture, named blind UNET (BUNET), that enables input-hiding segmentation on medical images. In the proposed protocol, we use a combination of cryptographic building blocks to ensure that client-side encryption is enforced on all data related to the patients, and that practical inference time can also be achieved. As a result, medical institutions can take advantage of third-party machine learning service providers without violating privacy regulations. Our code is available at Github: `https://github.com/dewenzeng/IEEE-Hackathon-Proposal`.

2 Use Scenario

We use the application of ML-based CT-based COVID-19 diagnosis. In this scenario, we suppose Alice (a doctor in a hospital) is the client who will provide the patients' health records such as CT scan, age, gender to Bob (a third-party machine-learning service providers) for medical analysis. These patient data will be used to analyze and infer the detection of COVID-19 and quantify the infectious region. The client can get to know whether this patient has COVID-19 as well as the infectious progressive stage.

3 Privacy Preserving

We provide a level-3 privacy preserving protocol, in which we suppose that the service provider and service users do not trust each other. Both parties follow the protocol prescribed above (e.g., encrypting real data, etc.), but want to learn as much information as possible from the other party. In order to guarantees that Alice learns only the segmentation results, while Bob learns nothing about the inputs from Alice, we efficiently use different cryptographic primitives such as homomorphic encryption and garbled circuits (GC) to design a complete secure protocol.

4 Technologies

4.1 Encryption Primitives

In this project, four types of cryptographic primitives are applied to prevent information leakage for both client and server, which include lattice-based packed additive homomorphic encryption (PAHE) [3, 4], additive secret sharing (ASS) [5], garbled circuits (GC) [9], and multiplication triples (MT) schemes [1, 7]. We will briefly introduce those four primitives in the following.

PAHE: In this cryptosystem, encryption (**Enc**) and decryption (**Dec**) functions perform group homomorphism between plaintext and ciphertext, that is preserving correctness of arithmetic operation after encryption. Our PAHE scheme contains the following three abstract operators, addition, and rotation. We use $[x]$ to denote the encrypted ciphertext of $x \in \mathbb{Z}^n$, and n represents the maximum number of plaintext integers that can be held in a single ciphertext.

Homomorphic addition ($\boxplus$): for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, $\mathbf{Dec}([\mathbf{x}] \boxplus [\mathbf{y}]) = \mathbf{x} + \mathbf{y}$. Note that it is the same for homomorphic subtraction, $\mathbf{Dec}([\mathbf{x}] \boxminus [\mathbf{y}]) = \mathbf{x} - \mathbf{y}$.

Homomorphic Hadamard product ($\boxdot$): for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, $\mathbf{Dec}([\mathbf{x}] \boxdot [\mathbf{y}]) = \mathbf{x} \circ \mathbf{y}$, where $\circ$ denotes the element-wise multiplication operator.

Homomorphic rotation (**rot**): for $\mathbf{x} \in \mathbb{Z}^n$, let $\mathbf{x} = (x_0, x_1, ..., x_{n-1})$, $\mathrm{rot}([\mathbf{x}], k) = (x_k, x_{k+1}, ..., x_{n-1}, x_0, ..., x_{k-1})$

ASS and Homomorphic Secret Sharing: A two-party ASS scheme consists of two operators, (**Share**, **Rec**), and some prime modulus $p_A \in \mathbb{Z}$. Each operator takes two inputs, where we have $\mathbf{s}_A = \mathbf{Share}(\mathbf{x}, \mathbf{s}_B) = (\mathbf{x} - \mathbf{s}_B) \bmod p_A$ and $\mathbf{x} = \mathbf{Rec}(\mathbf{s}_A, \mathbf{s}_B) = (\mathbf{s}_A + \mathbf{s}_B) \bmod p_A$. In [6], homomorphic secret sharing (HSS) is adopted, where ASS operates over on encrypted $\mathbf{x}$. For HSS, we have that

$$[\mathbf{s}_A] = \mathbf{Share}([\mathbf{x}], \mathbf{s}_B) = ([\mathbf{x}] \boxminus \mathbf{s}_B) \bmod p_A \tag{1}$$

$$[\mathbf{x}] = \mathbf{Rec}([\mathbf{s}_A], \mathbf{s}_B) = (\mathbf{s}_A \boxplus \mathbf{s}_B) \bmod p_A. \tag{2}$$

GC: GC can be considered as a more general form of homomorphic encryption. In particular, the circuit garbler, Alice, "encrypts" some function f along with her input x to Bob, the circuit evaluator. Bob evaluates $f(x, y)$ using his

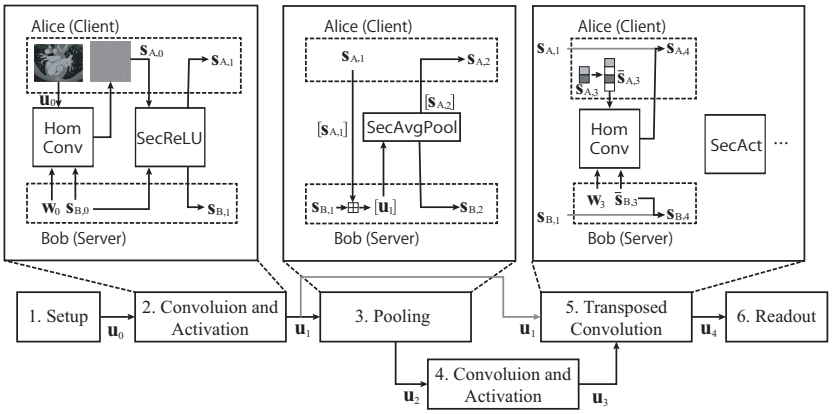

Figure 1: Illustration of the BUNET protocol

encrypted input y that is received from Alice obliviously, and obtains the encrypted outputs. Alice and Bob jointly "decrypt" the output of the function $f(x, y)$ and one of the two parties learns the result

MT: Beaver's MT [1] is a technique that performs multiplication on a pair of secret-shared vectors $\mathbf{x} = \mathbf{Rec}(\mathbf{s}_{A,x}, \mathbf{s}_{B,x})$ and $\mathbf{y} = \mathbf{Rec}(\mathbf{s}_{A,y}, \mathbf{s}_{B,y})$ between Alice and Bob. Here, we take computations performed by Alice as an example, and only note that the exact same procedure is also executed by Bob on his shares of secrets. To compute $\mathbf{x} \circ \mathbf{y}$, Alice and Bob first pre-share a set of respective multiplication triples $(\mathbf{a}_A, \mathbf{b}_A, \mathbf{c}_A)$ and $(\mathbf{a}_B, \mathbf{b}_B, \mathbf{c}_B)$, where we have $\mathbf{Rec}(\mathbf{a}_A, \mathbf{a}_B) \circ \mathbf{Rec}(\mathbf{b}_A, \mathbf{b}_B) = \mathbf{Rec}(\mathbf{c}_A, \mathbf{c}_B)$. Alice locally calculates $\mathbf{d}_A = \mathbf{s}_{A,x} \circ \mathbf{a}_A \mathrm{mod} p_A$, $\mathbf{e}_A = \mathbf{b}_{A,x} \circ \mathbf{s}_{A,y} \mathrm{mod} p_A$. Then, Alice and Bob publish their results $\mathbf{d}_A$,$\mathbf{d}_B$ and $\mathbf{e}_A$,$\mathbf{e}_B$. Finally, Alice obtains

$$\mathbf{g}_A = (\mathbf{c}_A + (\mathbf{e} \circ \mathbf{s}_{A,0,i}) + (\mathbf{d} \circ \mathbf{s}_{A,1,i} - \mathbf{d} \circ \mathbf{e}) \bmod p_A \tag{3}$$

where $\mathbf{d} = \mathbf{Rec}(\mathbf{d}_A, \mathbf{d}_B)$ and simliarly for $\mathbf{e}$. MT guarantees that $\mathbf{Rec}(\mathbf{g}_A, \mathbf{g}_B) = \mathbf{x} \circ \mathbf{y}$, where $\mathbf{g}_B$ is the MT results computed by Bob.

4.2 Protocol for Blind UNET

Figure 1 shows an example of BUNET protocol structure as a UNET architecture. For the first step, Alice converts the input image, either 2D or 3D, to one-dimensional vector $\mathbf{u}_0$. Bob does a similar transform on his filter weights to obtain one-dimensional vector $\mathbf{w}_0$. Regarding the convolutional layer, we conduct a standard (input-hiding) convolution on the input image with activation functions followed.

$$\mathbf{v}_0 = \mathrm{HomConv}(\mathbf{u}_0, \mathbf{w}_0), \text{ and } \mathbf{u}_1 = f_a(\mathbf{v}_0) = \mathrm{SecAct}(\mathbf{v}_0) \tag{4}$$

Here, f_a is some abstract activation function, which we choose between ReLU and square activation. While standard UNET architecture employs max pooling

as the downsampling method, we only use average pooling for less segmentation performance degradation. $\mathbf{u}_2 = \text{SecAvgPool}(\mathbf{u}_1)$. Then the encoding path are repetitions of the convolutional layer and pooling layer. As for the decoding path, protocol-level modifications are required for the image concatenation and padding operation. After Alice zero-pads $\mathbf{u}_3$ in an interleaving manner, $\overline{\mathbf{u}}_3$ is used as input to execute the following protocols.

$$\mathbf{v}_4 = \text{HomConv}(\overline{\mathbf{u}}_3, \mathbf{w}_3), \text{ and } \mathbf{u}_4 = \mathbf{v}_4 \parallel \mathbf{u}_1 \tag{5}$$

Lastly, we directly perform a secure Argmax without the Softmax operator, which can easily implemented by a GC protocol.

4.3 The Cryptographic Building Blocks

We introduce each of the cryptographic primitives used in the previous section.

HomConv: First, Alice performs an integer discrete Fourier transform (DFT) on $\mathbf{u}$ and obtain its frequency-domain representation, $\hat{\mathbf{u}}$. She then encrypts this input array into a ciphertext $[\hat{\mathbf{u}}]$ by running $[\hat{\mathbf{u}}] = \mathbf{Enc}_\kappa([\hat{\mathbf{u}}])$, where κ is the encryption key. The resulting ciphertext is transferred to Bob. At the same time, Bob applies DFT on his filter weights to obtain $\hat{\mathbf{w}}$. Then, Bob computes $[\hat{\mathbf{v}}] = [\hat{\mathbf{u}} \circ \hat{\mathbf{w}}] = [\hat{\mathbf{u}}] \boxdot \hat{\mathbf{w}}$. Finally, Bob appplies HSS as $[\hat{\mathbf{s}_A}] = \mathbf{Share}([\hat{\mathbf{v}}], \hat{\mathbf{s}}_B)$. Bob keeps $\hat{\mathbf{s}}_B$ and returns $[\hat{\mathbf{s}_A}]$ to Alice. Both Alice and Bob run inverse DFT on their shares of secrets and obtain $\mathbf{s}_A$ and $\mathbf{s}_B$, respectively.

SecAct: For **ReLU**, Alice first garbles the circuit with her share of secret $\mathbf{s}_A$. The garbled circuit computes the following function

$$\mathbf{v} = \mathbf{Rec}(\mathbf{s}_A, \mathbf{s}_B), \text{ and } \bar{\mathbf{s}}_A = \mathbf{Share}(\text{ReLU}(\mathbf{v}, \bar{\mathbf{s}}_B) \tag{6}$$

where $\bar{\mathbf{s}}_B$ is a freshly generated share of secret from Bob. On the other hand, we use MT on the square activation. First, we share the scret $\mathbf{v}$ twice among Alice and Bob. The MT protocol in Section 4.1 can then be executed, where both $\mathbf{x}$ and $\mathbf{y}$ equal $\mathbf{v}$. After that, Alce and Bob respectively obtain $\mathbf{g}_A$ and $\mathbf{g}_B$ where $\text{Rec}(\mathbf{g}_A, \mathbf{g}_B) = \mathbf{v}^2$.

SecPool: In [6], it is shown that max-pooling can be implemented using the GC protocol in Eq. 6, where we replace the ReLU operator with the MaxPool operator. Meanwhile, for secure average pooling, we can use a simple protocol based on PAHE. Specifically, we can compute the window-wise sum of some vector $\mathbf{v}$ by calculating $\text{SecAvgPool}(\mathbf{v}) = \sum_{i=0}^{w-1} \text{rot}([\mathbf{v}], i)$, where w is the pooling window size.

5 Experiment Result

In this section, we explore how the neural architecture of BUNET impact on the segmentation performance. In particular, it is important to see if the the proposed architectural modifications for UNET result in satisfactory prediction accuracy while accelerating the network inference time. The images in the

Table 1: The Dice Accuracy and Total Runtime of BUNET Protocol on the COVID-19 Dataset with Different Quantization Factors

	Float	32-bit	24-bit
Quantization		12-bit Int., 4-bit Frac.	8-bit Int., 4-bit Frac.
COVID-19 Dice	1.0	0.998	0.970
COVID-19 Time (s)	-	17,951	15,906

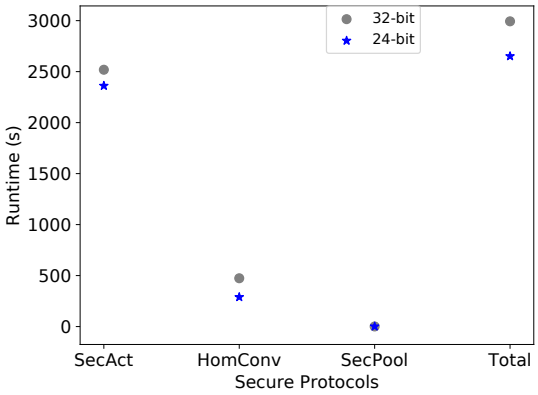

Figure 2: Runtime distribution for different cryptographic building blocks in a single run of secure segmentation.

COVID-19 dataset has $512 \times 512 \times 512$ input for binary segmentation. Table 1 summarizes the dice scores and runtime under two architectural settings. Here, the pooling function is average pooling unless otherwise stated. The UNET architecture is robust in low-quantization environment, where we see only a fraction of accuracy difference between floating point, 32-bit and 24-bit quantization factors. In particular, we see that giving more quantization bits to the integer part of the operands result in better accuracies.

We record the total runtime for a single blind image segmentation with respect to a UNET architecture with different quantization factors. In Fig. 2, we illustrate the runtime distribution for different secure protocols in BUNET. It is observed that, as demonstrated in previous works [8, 6, 2], the runtime for non-linear (i.e., ReLU) activations dominate the total runtime across architectures, while average pooling protocols are as light as a frequency-domain homomorphic convolution operations.

6 Conclusion

In this work, we propose BUNET to perform blind medical image segmentation on encrypted COVID-19 CT images. We apply a combination of cryptographic building blocks to ensure all patient data are encrypted in the data transfer and processing stage. Therefore, medical institutions can take advantage of third-

party machine learning service providers for COVID19 detection and analysis without violating privacy regulations.

References

[1] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Annual International Cryptology Conference*, pages 420–432. Springer, 1991.

[2] Song Bian, Tianchen Wang, Masayuki Hiromoto, Yiyu Shi, and Takashi Sato. ENSEI: Efficient secure inference via frequency-domain homomorphic convolution for privacy-preserving visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9403–9412, 2020.

[3] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[4] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[5] Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael Raskin. On the communication required for unconditionally secure multiplication. In *Annual International Cryptology Conference*, pages 459–488. Springer, 2016.

[6] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1651–1669, 2018.

[7] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making spdz great again. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 158–189. Springer, 2018.

[8] Jian Liu, Mika Juuti, Yao Lu, and N Asokan. Oblivious neural network predictions via MinioNN transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–631. ACM, 2017.

[9] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.