TOWARD SECURE IMAGE DENOISING: A MACHINE LEARNING BASED REALIZATION

Yifeng Zheng[*†], *Cong Wang*[*†], *and Jiantao Zhou*[‡]

[*]Department of Computer Science, City University of Hong Kong, Hong Kong
[†]City University of Hong Kong Shenzhen Research Institute, Shenzhen, 518057, China
[‡]Department of Computer and Information Science, University of Macau, Macau

ABSTRACT

Image denoising via machine learning techniques, particularly neural networks, has been shown to achieve state-of-the-art performance. However, in practice security and privacy issues undesirably arise in applying a trained machine learning model to image denoising. In this paper, we propose a system framework that enables the owner of a trained machine learning model to provide secure image denoising service to an authorized user, via the aid of cloud computing. Our framework ensures that the cloud server learns nothing about the model and the user's images, while the user learns nothing about the model except denoised images. Experiments are conducted for performance evaluation, and the results show that our design can achieve denoising quality close to that in the plaintext domain. For future work, we plan to explore various directions for optimizing the runtime performance.

Index Terms— Image denoising, machine learning, neural network, privacy, cloud computing

1. INTRODUCTION

Image denoising is a fundamental problem in the field of image processing. With a noisy image as input, an image denoising procedure aims to output a denoised image where the noise is reduced. Nowadays, it is a popular trend to learn an image denoising procedure via machine learning techniques, particularly neural networks, as this has shown state-of-the-art performance [1, 2, 3]. Typically, image denoising from neural networks includes two phases: (i) a training phase where a neural network model is trained from some training data consisting of pairs of noisy and noise-free images, and (ii) a denoising phase where the trained neural network model is applied to denoise noisy images.

This paper focuses on the denoising phase of neural network based image denoising. We start with considering a realistic setting in which a model owner has trained a neural network model and a user is interested in leveraging the model to denoise his own noisy images. For example, as training a model typically requires expertise in sophisticated parameter tuning, a hospital may seek to leverage a model already trained by a medical research center for the denoising of noisy medical images. Such an application via machine learning is an emerging service paradigm nowadays [4, 5].

To accommodate such a paradigm, we may trivially ask the model owner to directly give the neural network model to the user for local denoising, or to let the user submit the images to the model owner for remote denoising. However, such trivial mechanisms may not be easy to be enforced in practice due to security and privacy concerns. On the model owner side, he may not be willing to directly reveal the model as (i) the model may constitute a competitive advantage and thus should be kept confidential and (ii) the model may leak information about the training data which is potentially privacy-sensitive [5]. From the perspective of the user, the images, e.g., medical images and face images, may easily reveal privacy-sensitive information. Therefore, protection for both the neural network model of the model owner and the image of the user is highly demanded in applying the trained neural network model to image denoising.

A natural direction towards capturing the functionality and security of the above application scenario is secure two-party computation (e.g., [5]). However, this requires that the model owner and the user should be synchronous, which means that both of them need to be online at the same time upon the trigger of a denoising request. Besides, the model owner should be dedicated to both the training phase and the denoising phase, which imposes extra workload and resource investment that might not be desired by the model owner.

In light of the above observations, in this paper, we initiate the first study toward a secure framework for neural network based image denoising, where (i) the model owner and the client are made asynchronous and do not need to be online at the same time and (ii) the model owner can put the focus on maintaining and updating the neural network model, and get relieved of undesirable burden in the denoising phase. Specifically, our framework introduces a semi-trusted cloud server to obliviously aid the model owner to provide the denoising service to the authorized user. Such a server-aided model is quite natural and realistic given the proliferation of cloud ser-

This work was supported in part by the Research Grants Council of Hong Kong under Grant CityU C1008-16G, the National Natural Science Foundation of China under Grant 61572412, the Macau Science and Technology Development Fund under Grant FDCT/022/2017/A1, and a Microsoft Azure grant for research.

vices, and has also been widely adopted in the literature (e.g., [6, 7, 8]). Note that similar to prior work, we assume that the semi-trusted cloud server is non-colluding, which means that the cloud server will not collude with all the other parties. The intuition behind this assumption is that cloud service providers like Amazon and Google are well-established companies and there are little incentives for them to put the reputation at risk. Additional rationale includes the existence of audits and the fear of legal/financial repercussion [7, 8].

Our design ensures that along the workflow, the cloud server is oblivious to both the neural network model of the model owner and the image of the user, while the user does not learn the neural network model. While there are some prior works on secure image denoising via other techniques (e.g., [9, 10], just to list a few), secure neural network based denoising has not been explored before, to our best knowledge. Although the runtime performance is not yet attractive at this moment, we will provide comprehensive discussion on the existence of a large space where we can explore runtime performance optimization from various dimensions. In what follows, we will give our problem statement in Section 2 and the detailed design in Section 3. Then, we will provide some preliminary experiment results in Section 4, and give the conclusion and future work in Section 5.

2. PROBLEM STATEMENT

2.1. Neural network based image denoising

In this paper, we resort to the neural network model proposed in [1] as the first instantiation of our efforts toward secure neural network based denoising, due to its simplicity and promising performance. In [1], it is proposed that a multi-layer perceptron (MLP) can be used as the neural network model for high-quality denoising. Given an input vector $\mathbf{p}$, an MLP is essentially a nonlinear function that maps $\mathbf{p}$ to the output vector $\mathbf{q}$, via processing the input through several layers. For example, an MLP with $L-1$ hidden layers can be written as,

$$\mathbf{q} = \mathbf{W}_L \cdot tanh(...tanh(\mathbf{W}_1\mathbf{p} + \mathbf{b}_1)...) + \mathbf{b}_L), \quad (1)$$

where $\mathbf{W}_i$ denotes the weight matrix and $\mathbf{b}_i$ the bias vector, of the ith layer ($i \in [1, L]$). For image denoising, the input vector $\mathbf{x}$ is a noisy patch of the image to be denoised, and the output vector $\mathbf{y}$ is the denoised patch. So, given a noisy image, we first need to decompose it into overlapping patches and then apply the model to the denoising of each patch separately. Finally, the denoised image is obtained by placing the denoised patches at the locations of their noisy counterparts and averaging/weighting the overlapping region [1].

2.2. System model

Our work targets cloud-aided secure neural network based image denoising, which is illustrated in Fig. 1. We consider the setting where a model owner holding a trained neural network model allows the user to use the model for image denoising, with the aid of a cloud server. For security and privacy concerns, the model owner will send the trained neural network model in encrypted form to the cloud server. To leverage the model, the user is required to run a protocol with the cloud server for secure denoising over his noisy images.

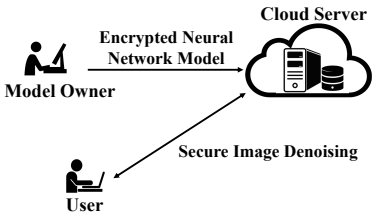

Fig. 1. Cloud-aided secure image denoising model.

2.3. Trust assumptions

We consider semi-honest adversaries in our system, which refers to the cloud server and the user. They will honestly follow our protocol specification, yet are interested in inferring private information that is out of their access rights. In particular, the cloud server tries to learn the values in $\mathbf{p}$, $\{\mathbf{W}_i\}_{i=1}^L$, $\{\mathbf{b}_i\}_{i=1}^L$, and $\mathbf{q}$, while the user attempts to learn the values in $\{\mathbf{W}_i\}_{i=1}^L$ and $\{\mathbf{b}_i\}_{i=1}^L$. Following prior works, we assume that the cloud server is non-colluding, i.e., there is no collusion between the cloud server and the user. Our security goal is to ensure that the cloud server learns nothing about the values in $\mathbf{p}$, $\{\mathbf{W}_i\}_{i=1}^L$, $\{\mathbf{b}_i\}_{i=1}^L$, and $\mathbf{q}$, and the user learns nothing about the values in $\{\mathbf{W}_i\}_{i=1}^L$ and $\{\mathbf{b}_i\}_{i=1}^L$. Similar to prior works on secure neural network based applications [5], we do not aim to protect the sizes of $\mathbf{p}$, $\mathbf{q}$, $\{\mathbf{W}_i\}_{i=1}^L$, and $\{\mathbf{b}_i\}_{i=1}^L$. Note that in principle the user can exploit the usage of the model as a blackbox oracle to launch model extraction attacks [11]. However, the cloud server can rate limit the user's requests to slow down or bound the leakage.

3. DESIGN OF SECURE NEURAL NETWORK BASED IMAGE DENOISING

3.1. Construction

Our construction for secure neural network based image denoising includes two phases. In phase 1, the model owner encrypts the neural network model and sends it to the cloud server. In phase 2, the user runs a protocol with the cloud server for secure denoising. Each phase is elaborated below.

Phase 1. To upload the neural network model to the cloud server, the model owner first encrypts each weight matrix and each bias vector. We use additively homomorphic cryptosystem for encryption, particularly the Paillier cryptosystem. Let (pk_u, sk_u) denote a key pair of the user of the Paillier cryptosystem, where pk_u is the public key and sk_u is the secret key. Also, let HE be the encryption algorithm of the Paillier cryptosystem. The model owner encrypts each element

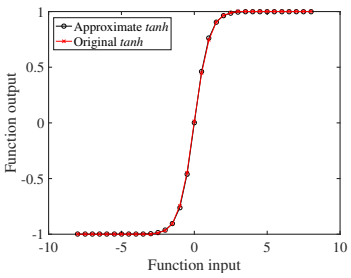

Fig. 2. Original $tanh(\cdot)$ function and its approximation.

of $\mathbf{W}_i$ and $\mathbf{b}_i$ under pk_u. We denote the element-wise encryption of $\mathbf{W}_i$ and $\mathbf{b}_i$ as $\mathsf{HE}_{pk_u}(\mathbf{W}_i)$ and $\mathsf{HE}_{pk_u}(\mathbf{b}_i)$, respectively. The model owner sends $\{\mathsf{HE}_{pk_u}(\mathbf{W}_i)\}_{i=1}^{L}$ and $\{\mathsf{HE}_{pk_u}(\mathbf{b}_i)\}_{i=1}^{L}$ to the cloud server.

Phase 2. We now describe how to securely denoise a noisy patch $\mathbf{p}$ of a noisy image. For simplicity of presentation, we will present our protocol for the case assuming one hidden layer. That is, $\mathbf{q} = \mathbf{W}_2 tanh(\mathbf{W}_1\mathbf{p} + \mathbf{b}_1) + \mathbf{b}_2$. The extension to more layers is quite natural, as the output of this layer serves as input of the next layer, followed by the same necessary operations.

Firstly, we need to somehow compute the element-wise encryption of $\mathbf{W}_1\mathbf{p}$, i.e., $\mathsf{HE}_{pk_u}(\mathbf{W}_1\mathbf{p})$, which requires encrytped multiplication. To this end, we resort to the protocol in [12], which enables the cloud server holding $\mathsf{HE}_{pk_u}(m_1)$ and $\mathsf{HE}_{pk_u}(m_2)$ to obtain $\mathsf{HE}_{pk_u}(m_1 \cdot m_2)$, while the user learns nothing. Through this protocol, we can enable the cloud server to obliviously obtain $\mathsf{HE}_{pk_u}(\mathbf{W}_1\mathbf{p})$, while the user learns nothing (except his own input $\mathbf{p}$). Subsequently, based on the property of homomorphic encryption, the cloud server can easily compute $\mathsf{HE}_{pk_u}(\mathbf{W}_1\mathbf{p} + \mathbf{b}_1)$.

After $\mathsf{HE}_{pk_u}(\mathbf{W}_1\mathbf{p} + \mathbf{b}_1)$ is obtained at the cloud server side, we then need to evaluate the activation function on each element of $\mathbf{W}_1\mathbf{p} + \mathbf{b}_1$ in the encrypted domain. However, the $tanh(\cdot)$ function requires exponentiation and division, which is hard to be efficiently computed in the ciphertext domain [5]. Instead of directly evaluating the $tanh(\cdot)$ function, our insight is to seek a secure-computation-friendly approximation that can be efficiently computed using secure computation techniques. We observe that the approximation in [13] is a good choice, which is a piecewise non-linear approximation of the $tanh(\cdot)$ function with an average error 4.1×10^{-3} and maximum error 2.2×10^{-2}. In particular, as plotted in Fig. 2, the $tanh(x)$ function is approximated as:

$$tanh(x)^* = \begin{cases} k \times (m_1|x|^2 + c_1|x| + d_1), 0 \le |x| \le a \\ k \times (m_2|x|^2 + c_2|x| + d_2), a < |x| \le b \\ k, \text{otherwise} \end{cases},$$

where k=$sign(x)$, and m_1, m_2, c_1, c_2, d_1, d_2, a and b are -0.2716, -0.0848, 1, 0.42654, 0.016, 0.4519, 1.52, and 2.57, respectively.

Given this approximation, we now show how to securely evaluate the approximate function $tanh(\cdot)^*$ on a given ciphertext $\mathsf{HE}_{pk_u}(x)$. Based on Yao's garbled circuits, we design a secure evaluation protocol which enables the cloud server holding $\mathsf{HE}_{pk_u}(x)$ to obtain $\mathsf{HE}_{pk_u}(tanh(x)^*)$, while the user learns nothing. Our protocol is as follows:

1. The cloud server chooses a random mask r and produces $\mathsf{HE}_{pk_u}(x + r)$, which is sent to the user.
2. The user performs decryption to obtain $x+r$. Then, the user builds a garbled circuit, which takes as input the garbled values of $x + r$, r, and r_u. Here r_u is a random mask chosen by the user. Inside the garbled circuit, the original value x is first recovered via $(x + r - r)$and then the $tanh(x)^*$ is computed. To protect the result, the garbled circuit will output $tanh(x)^* + r_u$ instead of $tanh(x)^*$.
3. After the garbled circuit is built, the user sends it along with the garbled values of $x + r$, r_u to the cloud server. Besides, $\mathsf{HE}_{pk_u}(r_u)$ is also sent, which will be used to transform the result $(tanh(x)^* + r_u)$ to $\mathsf{HE}_{pk_u}(tanh(x)^*)$ so that it can be used in subsequent computation.
4. The cloud server runs a 1-out-of-2 oblivious transfer protocol with the user to obtain the garbled value of r, and then evaluates the garbled circuit. The evaluation outputs $tanh(x)^* + r_u$. Then, the cloud server computes $\mathsf{HE}_{pk_u}(tanh(x)^*)$ via $\mathsf{HE}_{pk_u}(tanh(x)^* + r_u) \cdot \mathsf{HE}_{pk_u}(r_u)^{(-1)}$.

Based on this secure evaluation protocol, the cloud server can obtain $\mathsf{HE}_{pk_u}(tanh(\mathbf{W}_1\mathbf{p}+\mathbf{b}_1)^*)$, i.e., the element-wise encryption of $tanh(\mathbf{W}_1\mathbf{p} + \mathbf{b}_1)^*$. The remaining operations to obtain $\mathsf{HE}_{pk_u}(\mathbf{W}_2 tanh(\mathbf{W}_1\mathbf{p} + \mathbf{b}_1)^* + \mathbf{b}_2)$ at the cloud server side is straightforward, which only needs encrypted multiplication and encrypted addition as before. After $\mathsf{HE}_{pk_u}(\mathbf{W}_2 tanh(\mathbf{W}_1\mathbf{p}+\mathbf{b}_1)^*+\mathbf{b}_2)$ is obtained, the cloud server can send it to the user, who then performs decryption to obtain $\mathbf{W}_2 tanh(\mathbf{W}_1\mathbf{p} + \mathbf{b}_1)^* + \mathbf{b}_2$.

3.2. Security guarantees

Our security design can ensure that the cloud server learns nothing about the values in $\mathbf{p}$, $\mathbf{W}_i$, $\mathbf{b}_i$, and $\mathbf{q}$, while the user only learns the denoised patch $\mathbf{q}$ without knowing $\mathbf{W}_i$ and $\mathbf{b}_i$. Such security guarantees can be easily understood by investigating the workflow of our design, according to the modular sequential composition theorem [14, 4]. Firstly, the cloud server receives encrypted $\mathbf{W}_i$ and $\mathbf{b}_i$ under the semantically-secure Paillier cryptosystem, for which it does not have the key for decryption. Then, during the denoising procedure, there are some interactions among the cloud server and the user. Recall that these interactions are realized via the encrypted multiplication protocol and the garbled circuit based secure evaluation protocol. The security properties of these protocols ensure that the interactions reveal nothing about the underlying private information to the cloud server/user. Therefore, in our security design the cloud server learns nothing while the user only learns the denoised result.

Fig. 3. Five test images.

(a) (b) (c)

Fig. 4. An example of visual denoising result. (a) Noisy image; (b) Baseline result; (c) Our result.

4. EXPERIMENTS

4.1. Experiment setup

Our experiments are conducted on Microsoft Azure standard D12 instance (with 4 cores, 28GB RAM, and the Ubuntu Server 16.04 LTS system) using Java. For additively homomorphic encryption, we use the Paillier cryptosystem[1]. For garbled circuits, we use the ObliVM-lang programming framework [15] for proof-of-concept implementation. We use the trained neural network model[2] in [16] for test, which is an enhanced result on [1]. This model has four hidden layers, of which the sizes are 3072, 3072, 2559, and 2047, respectively. The output layer has a size 289, i.e., a denoised patch with size 17×17. We use five popular standard test images (with size 512×512) in our experiments, which is shown in Fig. 3. For demonstration purpose, the test images are corrupted with Gaussian noise with standard deviation 25. Each test noisy image is decomposed into overlapping patches of size 17×17 with a stride size 3. To denoise a noisy patch $\mathbf{p}$, the neural network model takes as input a noisy patch $\mathbf{p}^*$ of size 39×39, which includes not only the corresponding noisy pixels of $\mathbf{p}$ but also the surrounding ones. Following [1], each noisy pixel q is normalized via $(q/255 - 0.5) \cdot 0.2$, before being fed to the neural network.

4.2. Performance evaluation

We first compare the denoising quality of our design with that of the plaintext baseline method. Fig. 4 shows a visual example of the denoising result. The PSNR of the noisy image is 20.16 dB, while the PSNRs from the baseline method and our security design are 32.25 dB and 32.11 dB, respectively. The gap is 0.14 dB. Table 1 summarizes the PSNR results for all test images. The average gap is 0.116 dB, indicating that our design achieves the denoising quality close to the baseline.

We now report some results on the computation side. In particular, we focus on measuring the computation cost of two core atomic operations, i.e., encrypted multiplication and secure evaluation of the approximate hyperbolic tangent function. In our test, for the encrypted multiplication, it takes about 56.85 ms on the user side and 68.8 ms on the cloud server side, respectively. For the protocol of secure evaluation of the $tanh(\cdot)^*$ function, the computation cost turns out to be about 230.09 ms on the user side and 237.41 ms on the cloud side. Denoising a noisy patch can take up to tens of hours, without implementation-wise optimization. Although not yet efficient at this moment, our construction provides the first feasible solution for secure neural network based image denoising. We leave runtime performance speedup to our future work, which will be discussed shortly.

Table 1. Comparison of PSNR (dB) results for test images

Image	Baseline	Ours	Gap
Lena	32.25	32.11	0.14
Monarch	31.7	31.57	0.13
Airplane	31.72	31.53	0.19
Man	29.81	29.73	0.08
Baboon	25.84	25.8	0.04

5. CONCLUSION AND FUTURE WORK

We presented the first framework toward secure neural network based image denoising, which allows the model owner of a trained neural network model to provide secure denoising service to the authorized user via the aid of cloud computing. The model owner is only required to send the encrypted neural network model to the cloud server and does not need to stay online during the secure denoising phase. Our design leverages the cryptographic techniques of homomorphic encryption and garbled circuits, ensuring that the cloud server learns nothing while the user only learns the denoised images. Experiments show that the denoising quality provided by our secure design is comparable to that in the plaintext domain.

One interesting direction for future work is to optimize the overall runtime performance, which could be investigated from various aspects. Firstly, as the core cryptographic operations (i.e., encrypted multiplication and secure evaluation of the approximate activation function) are performed element-wise, exploiting parallel computation on powerful GPU [3] is promising. Secondly, one may seek simpler polynomial approximation of the activation function so as to simplify the garbled circuit, and may consider packing multiple plaintext values in a single ciphertext [17] for computation. Thirdly, emerging trusted hardware like Intel SGX also provides opportunities to achieve minimal performance overhead [18]. Additionally, one may consider the substitute of homomorphic encryption like secret sharing, which is lightweight (yet may require multiple cloud servers). A second future direction is to enrich the functionality via extending the framework to embrace more complex neural network models (e.g., [3]), and via exploring how to also protect data privacy in the phase of training the neural network model.

[1]Paillier lib.: http://www.csee.umbc.edu/ kunliu1/research/Paillier.html

[2]Trained model: http://people.tuebingen.mpg.de/burger/neural_denoising/

6. REFERENCES

[1] Harold Christopher Burger, Christian J. Schuler, and Stefan Harmeling, "Image denoising: Can plain neural networks compete with bm3d?," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2392–2399.

[2] Junyuan Xie, Linli Xu, and Enhong Chen, "Image denoising and inpainting with deep neural networks," in *Proc. of Annual Conference on Neural Information Processing Systems*, 2012, pp. 350–358.

[3] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[4] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser, "Machine learning classification over encrypted data," in *Proc. of Network and Distributed System Security Symposium*, 2015, pp. 1–14.

[5] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proc. of ACM Conference on Computer and Communications Security*, 2017, pp. 99–99.

[6] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proc. of Symposium on Theory of Computing Conference*, 2012, pp. 1219–1234.

[7] Huang Lin, Jun Shao, Chi Zhang, and Yuguang Fang, "CAM: cloud-assisted privacy preserving mobile health monitoring," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 985–997, 2013.

[8] Payman Mohassel, Ostap Orobets, and Ben Riva, "Efficient server-aided 2pc for mobile phones," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 2, pp. 82–99, 2016.

[9] Xianjun Hu, Weiming Zhang, Ke Li, Honggang Hu, and Nenghai Yu, "Secure nonlocal denoising in outsourced images," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 12, no. 3, pp. 40:1–40:23, 2016.

[10] Yifeng Zheng, Helei Cui, Cong Wang, and Jiantao Zhou, "Privacy-preserving image denoising from external cloud databases," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1285–1298, 2017.

[11] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart, "Stealing machine learning models via prediction apis," in *Proc. of USENIX Security Symposium*, 2016, pp. 601–618.

[12] Zekeriya Erkin, Thijs Veugen, Tomas Toft, and Reginald L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1053–1066, 2012.

[13] Che-Wei Lin and Jeen-Shing Wang, "A digital circuit design of hyperbolic tangent sigmoid function for neural networks," in *Proc. of International Symposium on Circuits and Systems*, 2008, pp. 856–859.

[14] Ran Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.

[15] Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi, "Oblivm: A programming framework for secure computation," in *Proc. of IEEE Symposium on Security and Privacy*, 2015, pp. 359–376.

[16] Harold Christopher Burger, Christian J. Schuler, and Stefan Harmeling, "Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds," *CoRR*, vol. abs/1211.1544, 2012.

[17] Trinabh Gupta, Henrique Fingler, Lorenzo Alvisi, and Michael Walfish, "Pretzel: Email encryption and provider-supplied functions are compatible," in *Proc. of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 169–182.

[18] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa, "Oblivious multi-party machine learning on trusted processors," in *Proc. of USENIX Security Symposium*, 2016, pp. 619–636.