C^{2}PI: An Efficient Crypto-Clear Two-Party Neural Network Private Inference

Yuke Zhang[1], Dake Chen[1], Souvik Kundu[2], Haomei Liu[1], Ruiheng Peng[1], and Peter A. Beerel[1]
[1]Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA
[2]Intel Labs, San Diego, USA
[1]{yukezhan, dakechen, haomeili, ruihengp, pabeerel}@usc.edu, [2]souvikk.kundu@intel.com

***Abstract*—Recently, private inference (PI) has addressed the rising concern over data and model privacy in machine learning inference as a service. However, existing PI frameworks suffer from high computational and communication costs due to the expensive multi-party computation (MPC) protocols. Existing literature has developed lighter MPC protocols to yield more efficient PI schemes. We, in contrast, propose to lighten them by introducing an empirically-defined privacy evaluation. To that end, we reformulate the threat model of PI and use inference data privacy attacks (IDPAs) to evaluate data privacy. We then present an enhanced IDPA, named distillation-based inverse-network attack (DINA), for improved privacy evaluation. Finally, we leverage the findings from DINA and propose `C`^{2}`PI`, a two-party PI framework presenting an efficient partitioning of the neural network model and requiring only the initial few layers to be performed with MPC protocols. Based on our experimental evaluations, relaxing the formal data privacy guarantees `C`^{2}`PI` can speed up existing PI frameworks, including Delphi [1] and Cheetah [2], up to $2.89\times$ and $3.88\times$ under LAN and WAN settings, respectively, and save up to $2.75\times$ communication costs.**

I. Introduction

With the increasing complexity of deep neural network (DNN) models and their incredible training cost, machine learning inference as a service (MLaaS) has become an inevitable solution saving significant time, cost, and effort in democratizing ML services, even for non-experts [3]. However, increasing privacy concerns challenge the inference procedure when a client and a server hold the inference input and network separately and do not want to reveal their private properties to each other. The client could be a patient with sensitive medical data or a homeowner with private images, and the server could be a hospital system or a commercial company holding proprietary trained DNN models.

Private inference (PI) has appeared to address the privacy issue in MLaaS. Existing PI frameworks [1], [2], [4]–[13] leverage secret sharing (SS) and multiparty computation (MPC) protocols to enable the participants to jointly perform inference without revealing their input and model parameters to each other. Different from prior PI frameworks [1], [2], [4]–[13] where data privacy is formally preserved through cryptographic guarantees, we adopt an empirically-defined client data privacy model [14]–[16] to relax PI. In particular, the client's data privacy is defined based on the potential success of inference data privacy attacks (IDPAs) [14], [15]. Specifically, if IDPAs **cannot** recover the client's input data, the client's data privacy is deemed preserved. IDPAs are typically evaluated through the structural similarity index (SSIM) [17] that measures the human perceptual similarity of two images by considering the luminance, contrast, and structure of two images. Users can set an SSIM value (usually 0.3 [15]) as IDPA's failure threshold, namely, an SSIM below the threshold indicating a failed recovery. The introduction of IDPA-based privacy enables a finer-grained means of quantifying client's data privacy than the Boolean characterization associated with cryptographic protocols.

The inability to recover the client's input with accessible layer outputs implies that the neural network unintentionally preserves the client's input privacy, intuitively due to the irreversibility of the network. To investigate the potential for revealing the input from only later layers' activations, we first propose an improved IDPA, i.e., a distillation-based inverse-network attack (DINA) for an improved evaluation as opposed to the baseline IDPAs [14], [15].

Our attack results indicate that the server indeed often cannot disclose the client's input even if it obtains later layers' outputs. Based on this observation, we propose a novel two-party PI framework, namely, crypto-clear private inference (`C`^{2}`PI`), and relax the computational burden of existing PI methods from a new perspective. Specifically, `C`^{2}`PI` searches for a *boundary layer* in a model, after which the two parties no longer need the cryptographic primitives to preserve the client's SSIM-based data privacy. This allows the server to independently operate on the remaining layers with significantly lower computation and latency. We name the layers before and after the boundary layer as *crypto layers* and *clear layers*, respectively, with the boundary layer as the last crypto layer. Furthermore, we leverage a noise-adding mechanism to further thwart the IDPAs and enhance clients' data privacy.

The benefit of our `C`^{2}`PI` is three-folded: *(a) It helps to reduce computational complexity of existing PI schemes. (b) It protects the architecture of the clear layers while existing PI frameworks leak the whole network architecture to the client [1], [2], [5], [8].* It is worth mentioning that the carefully designed network architectures are typically considered as intellectual property of network owners [18]. *(c) The introduced fine-grained privacy quantification enables users to trade-off PI complexity with the guaranteed level of client's data privacy by tuning the IDPA's failure threshold.* Existing

This work was supported in part by the Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-1-7817.

PI frameworks can be considered a special case of $\texttt{C}^2\texttt{PI}$ where the boundary is at the last layer. We summarize our contributions as follows.

- We propose a distillation-based IDPA (DINA), forming an enhanced evaluation of client's data privacy. DINA outperforms existing alternatives [14], [15] by achieving $\sim 0.1-0.23$ more structural similarity (SSIM) in image recovery tasks.
- We propose an efficient two-party PI framework, $\texttt{C}^2\texttt{PI}$. To the best of our knowledge, this is the first effort to protect partial neural network architecture, and find the portion of a network that is not necessarily performed with heavy MPC protocols to maintain IDPA-based data privacy. In $\texttt{C}^2\texttt{PI}$, we leverage DINA to find a reliable and conservative boundary between crypto layers and clear layers. Moreover, $\texttt{C}^2\texttt{PI}$ is orthogonal to recent PI-lightening techniques [2], [8], [19], [20]. Our extensive experimental evaluations show that, $\texttt{C}^2\texttt{PI}$ can achieve $1.1\times - 1.82\times$ speedup, and save $\sim 2.5\times$ communication costs for the state-of-the-art two-party PI framework Cheetah [2].

II. Preliminaries

Notations. Given a pre-trained neural network model $\mathbf{M}$ held by the server and an inference input $\mathbf{x}$ held by the client, we denote the output of the first l layers and the inference output as $\mathbf{M}_l(\mathbf{x})$ and $\mathbf{M}(\mathbf{x})$, respectively. As the boundary layer can be after either a linear operation or a ReLU operation, we use decimal .5 to denote the ReLU operation. For example, layer 3 and layer 3.5 refer to the linear operation and ReLU operation in layer 3, respectively.

Threat model. In this work, we follow the semi-honest threat model, where both parties strictly follow the cryptographic protocols, but try to reveal their collaborator's private input by inspecting the information they received. In $\texttt{C}^2\texttt{PI}$, the server is allowed to get the outputs of the boundary and clear layers, from which server will try to recover client's input using IDPAs.

Inference data privacy attacks (IDPAs). The inference data privacy was first systematically studied in [15] for collaborative inference in split learning (SL), where a network $\mathbf{M}$ is split into two parts: $\mathbf{M^1}$ containing the first consecutive layers in $\mathbf{M}$ and $\mathbf{M^2}$ containing the remaining layers. Two participants, edge and cloud, hold $\mathbf{M^1}$ and $\mathbf{M^2}$ respectively. When performing the inference, the edge feeds its input $\mathbf{x}$ into $\mathbf{M^1}$ and sends the result $\mathbf{M^1}(\mathbf{x})$ to the cloud. The cloud then processes $\mathbf{M^2}(\mathbf{M^1}(\mathbf{x}))$ and shares the inference results with the edge if necessary. In the edge-cloud scenario, the cloud is curious-but-honest trying to recover edge's input $\mathbf{x}$ from $\mathbf{M^1}(\mathbf{x})$ through two kinds of IDPA, i.e., the *maximum likelihood attack (MLA)* and the *inverse-network attack (INA)* [15]. This threat model is suitable for our client-server scenario in the way that $\mathbf{M^1}$ and $\mathbf{M^2}$ are composed of our crypto layers and clear layers, respectively. Therefore, IDPAs can also be used to evaluate the privacy of our client's input. Assuming that the boundary in $\texttt{C}^2\texttt{PI}$ is layer l, we have $\mathbf{M^1}(\mathbf{x}) = \mathbf{M}_l(\mathbf{x})$. Despite the similarities between our client-server scenario and the edge-cloud scenario in SL, i.e., (1) we both partition a network into two parts, (2) client and edge need to pass their outputs to server and cloud, respectively, there is a distinction between the two settings, i.e., $\mathbf{M^1}$ is held by the server in the client-server scenario, whereas $\mathbf{M^1}$ is held by the edge in SL. Therefore, $\texttt{C}^2\texttt{PI}$ and SL address the privacy issue in different situations. In $\texttt{C}^2\texttt{PI}$, the server trains the network and provides service based on its property. In SL, the edge users hold both inference input and a pretrained network but want to move some inference processes to the server because of the limited computation and storage capacities at the edge.

MLA recovers the input $\mathbf{x}$ by solving an optimization problem $\hat{\mathbf{x}} = \texttt{argmin}_{\hat{\mathbf{x}}} \|\mathbf{M}_l(\hat{\mathbf{x}}) - \mathbf{M}_l(\mathbf{x})\|_2^2$ through gradient decent at a target layer l. INA constructs an inversion model $\mathbf{M}^*$ with a single architecture, trains the model by taking $\mathbf{M}_l(\mathbf{x}')(x' \in \text{TraningSet})$ and $\mathbf{x}'$ as the input and output, and recovers the input $\mathbf{x}$ by querying $\mathbf{M}^*$. Conceptually, this new model $\mathbf{M}^*$ approximates the inverse function of first l layers in $\mathbf{M}$. An enhanced INA (EINA) is proposed in [14] where the inversion model $\mathbf{M}^*$ consists of more powerful residual blocks [21].

Private Inference schemes. A series of works have been proposed for two-party PI in the past few years. CryptoNets [6] proposes a Homomorphic encryption (HE)-only approach, which requires changing the network structure and retraining the network with HE-friendly activation functions, such as the square function. MiniONN [4] first combines SS, Garbled Circuits (GCs) and linearly homomorphic encryption (LHE), to perform activation functions like ReLU without changing the network structure. Gazelle [5] optimizes the LHE techniques to speed up the inference runtime. XONN [7] leverages only GCs for binarized neural networks. Delphi [1] splits PI into offline (preprocessing) and online phases and moves most heavy cryptographic computations offline. CrypTFlow2 [8] proposes more efficient protocols for non-linear layers and division, yielding more than $20\times$ faster PI than Delphi. Cheetah [2], the state-of-the-art two-party PI strategy, presents an oblivious transfer (OT)-based protocol for non-linear operations and achieves $2\times - 5\times$ speedup than CrypTFlow2. Recently, PI protocols have also been proposed over the malicious client [11], [12], [22].

Privacy goals. In $\texttt{C}^2\texttt{PI}$, the client can only learn server's network architecture of the crypto layers, and the output of the inference. All parameters of server's network should be hidden. On the other hand, the server can learn only the outputs of the boundary and clear layers, as well as the inference output. Client's private input should not be revealed during operations of MPC protocols nor recovered by IDPAs. While $\texttt{C}^2\texttt{PI}$ achieves cryptographically formal guarantee on server's model parameter privacy, it targets the empirical SSIM-based protection on client's input data. Physical attacks such as side-channel attacks (SCAs) [18], [23] are out of the scope. However, the defenses against SCAs are possibly to be layered on top of our technique.

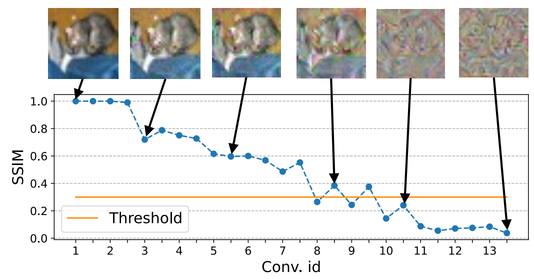
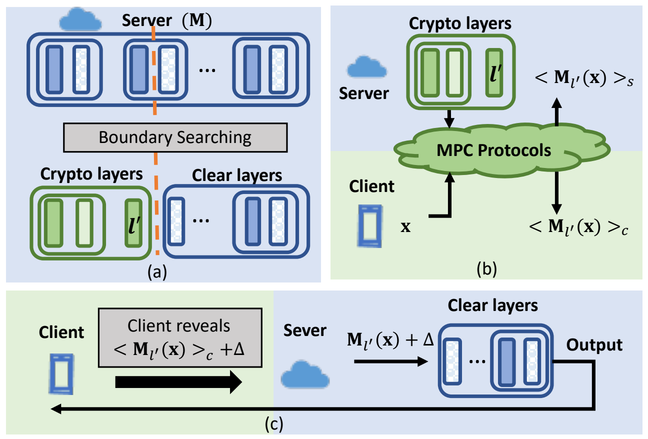


Fig. 1. MLA result on an image from CIFAR-10. When SSIM is below the threshold (0.3), the recovered image is difficult to identify.


Server (M)
Boundary Searching
Crypto layers
Clear layers
l'
(a)
Crypto layers
Server
$< \mathbf{M}_{l'}(\mathbf{x}) >_s$
MPC Protocols
Client
x
$< \mathbf{M}_{l'}(\mathbf{x}) >_c$
(b)
Client
Client reveals $< \mathbf{M}_{l'}(\mathbf{x}) >_c + \Delta$
Sever
$\mathbf{M}_{l'}(\mathbf{x}) + \Delta$
Clear layers
Output
(c)
Fig. 2. Framework of C^{2}PI: (a) Server identifies the boundary layer l'. (b) Operations in crypto layers are performed with MPC protocols (c) Client reveals its share to server. Then, server independently performs the operations in clear layers and shares the final result with the client.

III. C^{2}PI: Crypto-Clear Private Inference

We now show that the network naturally helps to hide the original input when the inference procedure enters into deeper layers.

Case study. We pretend to be the curious server and use MLA at each layer of a VGG16 [24] model, denoted as $\mathbf{M}$, to reconstruct the client's input $\mathbf{x}$, which is one image from CIFAR-10 [25]. When we target layer l^*, we assume that we have the intermediate outputs at layers $l \geq l^*$, i.e., $\mathbf{M}_l(\mathbf{x})(l \geq l^*)$, and we do not have the outputs at layers $l < l^*$, i.e., $\mathbf{M}_l(\mathbf{x})(l < l^*)$. The attack results are presented in Figure 1 where SSIM becomes less than 0.3 after layer 10. This experiment indicates that server cannot recover client's input from $\mathbf{M}_l(\mathbf{x})(l \geq 10)$ through MLA. Therefore, server could potentially be allowed to have layer outputs after layer 10 for a lighter PI procedure without violating the client's data privacy.

A. C^{2}PI *Framework*

Figure 2 shows the C^{2}PI framework. Firstly, the server searches for the boundary between the crypto and the clear layers following Algorithm 1. In the semi-honest threat model, a curious server will not deviate from this algorithm, and a third-party notary organization can be involved to ensure honesty. Then, server and client jointly perform the operations in crypto layers with a chosen two-party PI method, e.g.,

Algorithm 1 Crypto-Clear Boundary Searching

```
Input: A network model $\mathbf{M}$, the number of layers $n$, accuracy threshold $\delta$, noise magnitude $\lambda$, ssim-threshold $\sigma$
Output: Boundary layer id $l'$
1:  $l' = n - 1$
2:  $avg\_ssim = \text{IDPA}(l')$
3:  while $avg\_ssim < \sigma$ do
4:      $l' = l' - 1$
5:      $avg\_ssim = \text{IDPA}(l')$
6:  end while
7:  $l' = l' + 1$
8:  $n\_acc = accuracy(l', \lambda)$
9:  while $n\_acc < \delta$ do
10:     $l' = l' + 1$
11:     $n\_acc = accuracy(l', \lambda)$
12: end while
13: return $l'$
```

Delphi [1] or Cheetah [2]. At the end of PI, client and server each hold an additive share of the boundary layer's output. Client then adds uniform-distributed noise to its share and reveals the noised share to the server [26], [27]. Server searches the maximum noise magnitude in Algorithm 1 that yields an acceptable inference accuracy and deliver this to client before performing PI. After summing up the two shares, server performs the operations in clear layers on its own and reveals the inference output to the client at the end of C^{2}PI.

The boundary searching algorithm (Algorithm 1) contains two phases. During the first phase (line 1 to line 6), the server sweeps the layers from tail to the head of the model and applies IDPA to recover the inference input. Phase 1 terminates at layer l' at which IDPA begins to succeed in recovery. The potential boundary layer, i.e., the last crypto layer, is layer l'+1. Then, in the second phase (line 7 - line 11), server checks the accuracy by assuming the input of layer $l'+2$ becomes the noised input. If the accuracy is above an agreed threshold δ, $l' + 1$ is returned as the boundary layer. Otherwise, the server checks the layers after $l' + 1$ in sequence until obtaining a satisfying accuracy.

As the operations in crypto layers are performed with existing PI schemes, the data privacy of both parties at these layers is formally proved. Recalling that client's data privacy at clear layers depends on the quality of the IDPA, we propose a distillation-based inverse-network attack (DINA) to evaluate client's data privacy at clear layers and find the boundary layer.

B. *Distillation-Based Inverse Network Attack (DINA)*

Despite EINA [14] increasing the complexity of the inversion model to enhance its inverse ability, it does not take full advantage that server has access to the intermediate layer outputs of its own model, which can be used to guide the training of the inversion model. Therefore, we introduce distillation points in DINA to help the inversion model better approximate the target inverse function.

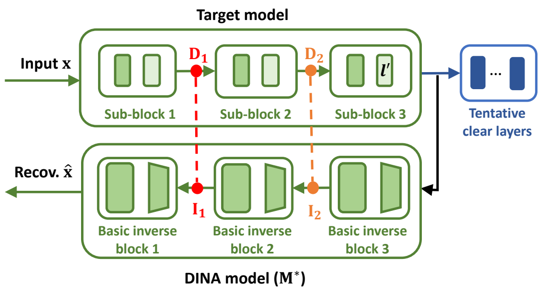


Fig. 3. Model architecture of DINA

Figure 3 presents the model architecture in DINA, which is composed of a sequence of *basic inverse blocks*. Each basic inverse block consists of a ResNet basic block [21] and a dilated convolution layer. Since the ReLU layer significantly affects the attack, we partition the tentative crypto layers before l' into sub-blocks that end with a ReLU layer, namely, each sub-block only contains one ReLU layer. The proposed attack then uses a basic inverse block to recover the input of one sub-block, as shown in Figure 3, each basic inverse block approximates the inverse function of the sub-block above it.

To better train each basic inverse block, DINA selects middle points between sub-blocks as distillation points [28] and applies a fine-grained distillation approach that optimizes the distance between the output of each basic inverse block and the feature map on the corresponding distillation point. The distances are incorporated into a new loss function:

$$\mathcal{L}_{DINA} = \sum_{j=1}^{N} \alpha_j \|\mathbf{D}_j - \mathbf{I}_j\|_2^2 + \alpha_0 \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \qquad (1)$$

where the first term is the weighted sum of distance terms at distillation points, α_j is the coefficient that controls the weight of the distance at distillation point j, $\mathbf{D}_j$ denotes the feature map at distillation point j in the target model, $\mathbf{I}_j$ is the input of basic inverse block j in DINA model, and N represents the total number of selected distillation points. The second term is the distance between the inference input $\mathbf{x}$ and the output $\hat{\mathbf{x}}$ from DINA model.

To assist a distillation point in providing effective guidance on its nearest basic inverse block, the attack applies monotonously increasing coefficients α_j from the output to input of DINA model: $\alpha_0 < \alpha_1 < \alpha_2 ... < \alpha_N$, this ensures that each basic inverse block obtains the most guidance from its nearest distillation point. In the example shown in Figure 3, there are two distillation points, colored in red and orange, respectively. Although the losses at the output of the DINA model and both distillation points contribute to optimizing parameters in the basic inverse block 3, the loss at the orange distillation point has the largest impact due to the monotonously increasing coefficients.

IV. Evaluations

Our experiments are conducted on AlexNet [25] and variants of VGG16/19 [24] [1]. We train these models on CIFAR-10 and CIFAR-100 [29] with an Nvidia A100 GPU.

[1]We use variant models with reduced channel size. The models are available at https://anonymous.4open.science/r/c2pi-256C

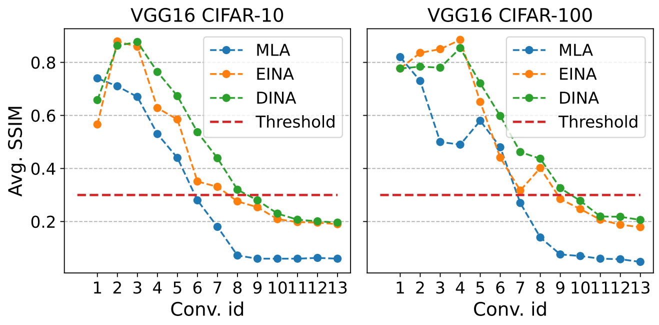


Fig. 4. Comparison of IDPAs including MLA, EINA, and DINA.

A. Comparison of IDPAs

We apply MLA [15], EINA [14], and DINA on each layer of VGG16 to recover images in CIFAR-10 and CIFAR-100 datasets. When targeting layer l, MLA solves $\hat{\mathbf{x}} = argmin_{\hat{\mathbf{x}}} \|\mathbf{M}_l(\hat{\mathbf{x}}) - \mathbf{M}_l(\mathbf{x})\|_2^2$ through gradient descent with 10000 iterations and randomly initialized $\hat{\mathbf{x}}$. In EINA, we construct an inversion network $\mathbf{M}^*$ with residual basic blocks [21] and train it using the loss function of $\mathcal{L}_{EINA} = \|\mathbf{x} - \mathbf{M}^*(\mathbf{M}_l(\mathbf{x}))\|_2^2$ and stochastic gradient descent optimizer. In DINA, we introduce distillation points and train $\mathbf{M}^*$ with the loss function in (1). The coefficients in our training are monotonously increasing as $\alpha_0 = 1, \alpha_1 = 3, \alpha_j = 2 * \alpha_{j-1} (j \geq 2)$. Both training processes are with a 0.001 learning rate. After training the model $\mathbf{M}^*$, we run the inference over 1000 images from each dataset and evaluate the recovery ability. The noise magnitude and the IDPA failure threshold for this experiment are 0.1 and 0.3, respectively.

Attack results are presented in Figure 4, where DINA achieves 0.229 and 0.205 more average SSIM than MLA at layer 7 on CIFAR-10 and CIFAR-100, respectively. DINA also presents 0.108 and 0.145 more SSIM than EINA at layer 7 on CIFAR-10 and CIFAR-100.

Recalling that Algorithm 1 first searches for a potential boundary layer after which IDPA begins to fail. MLA, EINA, and DINA return layers 7.5, 8.5, and 9 as the potential boundary layer for CIFAR-10, respectively, and layers 7.5, 9.5, and 10 for CIFAR-100, respectively. Therefore, DINA finds a more conservative boundary than MLA and EINA.

B. Choice of DINA's Loss Coefficients

In DINA, we use monotonously increasing coefficients $\alpha_j (j \geq 0)$ in the loss function for more effective guidance on the basic inverse blocks. In this section, we compare DINA with increasing coefficients $\alpha_0 = 1, \alpha_1 = 3, \alpha_j = 2 * \alpha_{j-1} (j \geq 2)$, denoted as DINA-c1, and DINA with uniform coefficients $\alpha_j = 1 (j \geq 0)$, denoted as DINA-c2. Figure 5 presents the attack results where DINA-c1 achieves a higher average SSIM. We use DINA-c1 in all of our experiments.

C. Effects of Adding Noise

Now we show that adding noise helps to thwart DINA. We use DINA to attack VGG16 on CIFAR-10 and CIFAR-100 with noise magnitude changing from 0 to 0.5. The attack results are presented in Figure 6 and we conclude that a

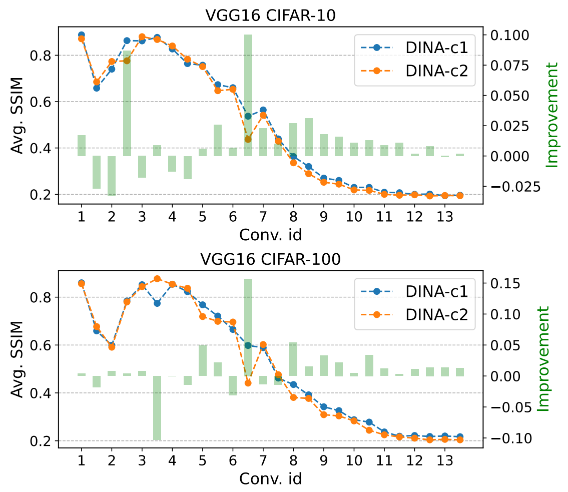


Fig. 5. Attack results of DINA-c1 and DINA-c2 on VGG16. The improvements are the increased average SSIM gained by DINA-c1.

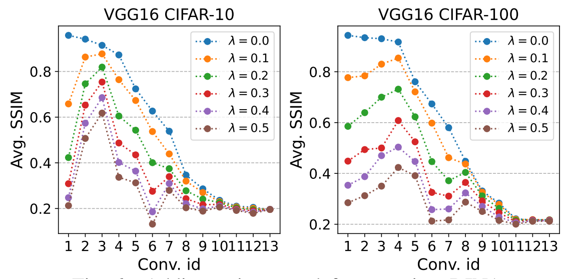


Fig. 6. Adding noise as a defense against DINA.

higher noise magnitude leads to a more vigorous defense against DINA and potentially results in an earlier boundary layer and more computational and communication savings. However, the inference accuracy degrades with the increasing noise magnitude as shown in Figure 7, where the accuracy at layer l is tested after feeding noised input to this layer. To balance the trade-off between the defense level and accuracy, we choose the noise magnitude of 0.1 in our experiments.

D. Find the Crypto-Clear Boundary

We apply the proposed boundary searching algorithm (Algorithm 1) to AlexNet and VGG16/19 on CIFAR-10 and CIFAR-100. In the first phase of the searching process, we find a potential boundary after which the server cannot recover client's input through DINA. We then check the accuracy with noise being added at these layers and decide the boundary as the earliest layer presenting less than 2.5% reduction in accuracy, a target that is similar to prior works [20], [30]. This searching procedure is presented in Figure 8. As users are free to tune the DINA failure threshold (σ), we show boundaries and accuracy corresponding to σ of 0.2 and 0.3 in Table I.

E. Computational Costs

Our implementations are built on top of Cheetah [2] and Delphi [1]. All the experiments in Table II are performed on Ubuntu 22.04 with 11GB of RAM. We run our benchmarks on two network settings, i.e., LAN and WAN. The network bandwidth and round-trip time are around 384MBps and 0.3ms in LAN, and 44MBps and 40ms in WAN [2].

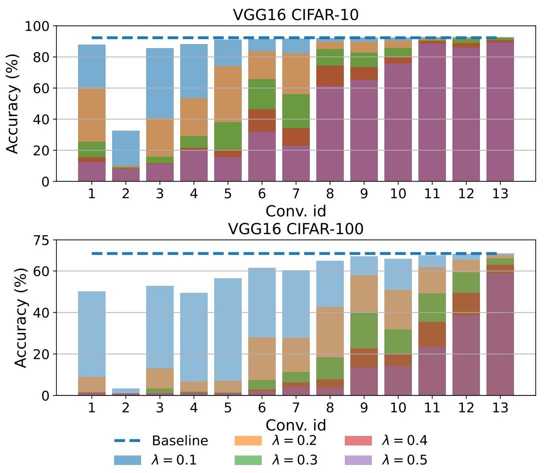


Fig. 7. The effect of adding noise on accuracy on CIFAR-10 and CIFAR-100.

TABLE I
C^2PI BOUNDARY AND ACCURACY

Dataset	Network	Full PI	C^2PI ($\sigma = 0.2$)		C^2PI ($\sigma = 0.3$)	
		Baseline Acc.	Boundary	Acc.	Boundary	Acc.
CIFAR-10	AlexNet	81.56	5	81.97	4	79.32
	VGG16	92.33	13.5	92.61	9	92.49
	VGG19	92.38	11	92.66	9	92.42
CIFAR-100	AlexNet	45.66	5	45.36	5	45.36
	VGG16	68.44	13.5	68.44	10	66.53
	VGG19	69.54	11	67.3	9	67.06

In Table II, we compare the full PI costs with C^2PI with two SSIM thresholds (0.2 and 0.3) on VGG16 and VGG19. When the crypto layers are performed with Delphi, C^2PI achieves up to 3.88$\times$ and 2.9$\times$ speedup for VGG16 and VGG19, respectively. When the crypto layers were performed with Cheetah, C^2PI was about 1.1 $\times$ $-$1.82$\times$ faster than full Cheetah, and requires $\sim$2.5$\times$ less communication. C^2PI ($\sigma = 0.2$) presents similar costs with full PI for VGG16 mainly because the boundary is quite late here.

V. Conclusion

In this paper, we propose an efficient two-party PI framework, C^2PI, which leverages IDPA-based privacy evaluation to relax existing PI methods. We also propose a powerful distillation-based IDPA, DINA, that can recover higher-quality images than its alternatives. In C^2PI, DINA is used to find a rigid boundary that guarantees the proposed IDPA-based privacy. Users can set the privacy level by tuning the DINA's failure threshold. Moreover, C^2PI can be applied to any existing two-party PI scheme over semi-honest threat model, and our experimental results indicate that C^2PI helps to reduce the computational costs of the state-of-the-art PI scheme.

Besides its significant role in C^2PI, DINA also helps address the privacy issue in split learning. We recognize that emerging IDPAs may throw a shadow on current C^2PI. However, we are glad to replace DINA with a more aggressive IDPA in C^2PI. On the other hand, we believe that cheap but effective countermeasures will appear accordingly and fortify C^2PI. Our future work includes exploring and applying more defenses against IDPA to preserve client's data privacy, and embedding C^2PI with PI methods that go beyond the semi-honest threat model, e.g., the malicious-client threat model.

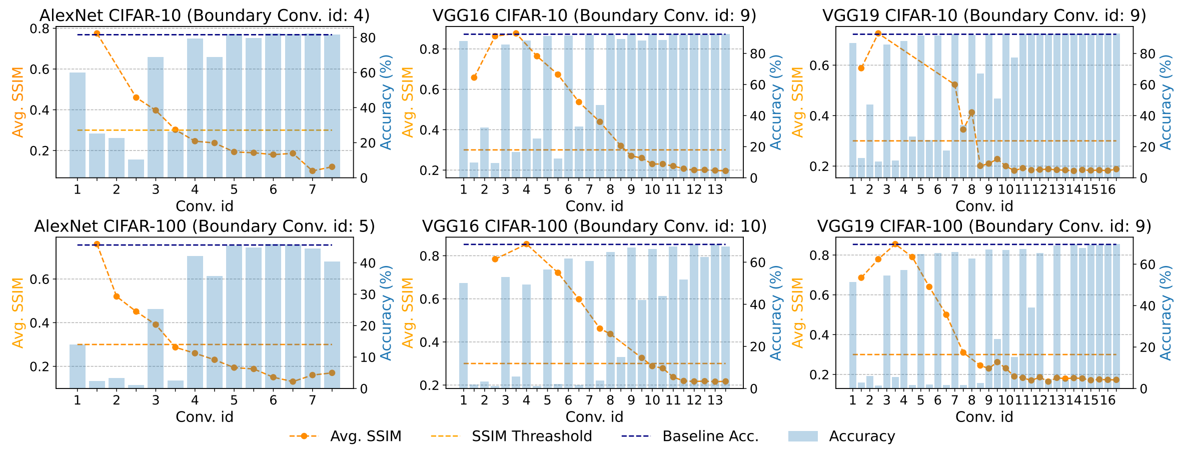


Fig. 8. Find the boundary for multiple models using DINA with 0.3 of failure threshold. Step 1: find a potential boundary after which the average SSIM begins to be below the threshold. Step 2: check the corresponding accuracy. Push the boundary at later layer until we obtain a satisfactory accuracy.

TABLE II
PERFORMANCE COMPARISON OF C^{2}PI AND DELPHI/CHEETAH ON CIFAR-10.

Network	Method	Full PI			C^{2}PI ($\sigma = 0.2$)			C^{2}PI ($\sigma = 0.3$)		
		Latency (s)		Commu. (MB)	Latency (s)		Commu. (MB) (Improv.)	Latency (s)		Commu. (MB) (Improv.)
		LAN	WAN		LAN (Improv.)	WAN (Improv.)		LAN (Improv.)	WAN (Improv.)	
VGG16	Delphi [1]	6166.47	9966.48	5163	6109.47 (~**1x**)	9869.12 (~**1x**)	5163 (~**1x**)	2351.5 (**2.62x**)	2568.45 (**3.88x**)	5143 (~**1x**)
	Cheetah [2]	13.72	25.27	179.64	14.38 (**1.19x**)	25.08 (**1x**)	163.8 (**1.1x**)	9.38 (**1.46x**)	14.76 (**1.71x**)	71.89 (**2.5x**)
VGG19	Delphi [1]	12780.36	13265.52	5184	5510.23 (**2.3x**)	6068.12 (**2.19x**)	5162 (~**1x**)	4409.95 (**2.9x**)	5373.34 (**2.47x**)	5143 (~**1x**)
	Cheetah [2]	16.81	27.67	211.4	11.89 (**1.51x**)	18.23 (**1.66x**)	89.55 (**2.39x**)	11.51 (**1.46x**)	15.23 (**1.82x**)	76.83 (**2.75x**)

REFERENCES

[1] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *29th USENIX Security Symposium (USENIX Security 20)*, Aug. 2020.
[2] Z. Huang, W. jie Lu, C. Hong, and J. Ding, "Cheetah: Lean and fast secure two-party deep neural network inference." Cryptology ePrint Archive, Paper 2022/207, 2022.
[3] S. Kundu, Q. Sun, Y. Fu, M. Pedram, and P. Beerel, "Analyzing the confidentiality of undistillable teachers in knowledge distillation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9181–9192, 2021.
[4] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 619–631, 2017.
[5] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "{GAZELLE}: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1651–1669, 2018.
[6] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International conference on machine learning*, pp. 201–210, PMLR, 2016.
[7] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "{XONN}:{XNOR-based} oblivious deep neural network inference," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1501–1518, 2019.
[8] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow2: Practical 2-party secure inference," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 325–342, 2020.
[9] S. Tan, B. Knott, Y. Tian, and D. J. Wu, "Cryptgpu: Fast privacy-preserving machine learning on the gpu," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1021–1038, IEEE, 2021.
[10] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4961–4973, 2021.
[11] G. Xu, X. Han, T. Zhang, H. Li, and R. H. Deng, "Simc 2.0: Improved secure ml inference against malicious clients," *arXiv preprint arXiv:2207.04637*, 2022.
[12] R. Lehmkuhl, P. Mishra, A. Srinivasan, and R. A. Popa, "Muse: Secure inference resilient to malicious clients," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2201–2218, 2021.
[13] L. Shen, Y. Dong, B. Fang, J. Shi, X. Wang, S. Pan, and R. Shi, "Abnn2: secure two-party arbitrary-bitwidth quantized neural network predictions," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 361–366, 2022.
[14] J. Li, A. S. Rakin, X. Chen, Z. He, D. Fan, and C. Chakrabarti, "Ressfl: A resistance transfer framework for defending model inversion attack in split federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10194–10202, 2022.
[15] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 148–162, 2019.
[16] Z. Liu, Z. Wu, C. Gan, L. Zhu, and S. Han, "Datamix: Efficient privacy-preserving edge-cloud inference," in *European Conference on Computer Vision*, pp. 578–595, 2020.
[17] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
[18] Y. Zhang, R. Yasaei, H. Chen, Z. Li, and M. A. Al Faruque, "Stealing neural network structure through remote fpga side-channel analysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4377–4388, 2021.
[19] Z. Ghodsi, N. K. Jha, B. Reagen, and S. Garg, "Circa: Stochastic relus for private deep learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2241–2252, 2021.
[20] M. Cho, A. Joshi, B. Reagen, S. Garg, and C. Hegde, "Selective network linearization for efficient private inference," in *International Conference on Machine Learning*, pp. 3947–3961, PMLR, 2022.
[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[22] N. Chandran, D. Gupta, S. L. B. Obbattu, and A. Shah, "Simc: Ml inference secure against malicious clients at semi-honest cost," *Cryptology ePrint Archive*, 2021.
[23] S. Maji, U. Banerjee, and A. P. Chandrakasan, "Leaky nets: Recovering embedded neural network models and inputs through simple power and timing side-channels—attacks and defenses," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12079–12092, 2021.
[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.
[26] N. D. Pham, A. Abuadbba, Y. Gao, T. K. Phan, and N. Chilamkurti, "Binarizing split learning for data privacy enhancement and computation reduction," *arXiv preprint arXiv:2206.04864*, 2022.
[27] T. Titcombe, A. J. Hall, P. Papadopoulos, and D. Romanini, "Practical defences against model inversion attacks for split neural networks," *arXiv preprint arXiv:2104.05743*, 2021.
[28] S. Kundu and S. Sundaresan, "Attentionlite: Towards efficient self-attention models for vision," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2225–2229, IEEE, 2021.
[29] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.
[30] S. Kundu, S. Lu, Y. Zhang, J. Liu, and P. A. Beerel, "Learning to linearize deep neural networks for secure and efficient private inference," *ICLR*, 2023.