Secure Medical Image Analysis with CrypTFlow

Javier Alvarez-Valle  
MSR UK  

Pratik Bhatu  
MSR India  

Nishanth Chandran  
MSR India  

Divya Gupta  
MSR India  

Aditya Nori  
MSR UK  

Aseem Rastogi  
MSR India  

Mayank Rathee  
MSR India  

Rahul Sharma  
MSR India  

Shubham Ugare  
UIUC, USA  

Abstract

We present CRYPFLOW, a system that converts TensorFlow inference code into Secure Multi-party Computation (MPC) protocols at the push of a button. To do this, we build two components. Our first component is an end-to-end compiler from TensorFlow to a variety of MPC protocols. The second component is an improved semi-honest 3-party protocol that provides significant speedups for inference. We empirically demonstrate the power of our system by showing the secure inference of real-world neural networks such as DENSENET121 for detection of lung diseases from chest X-ray images and 3D-UNet for segmentation in radiotherapy planning using CT images. In particular, this paper provides the first evaluation of secure segmentation of 3D images, a task that requires much more powerful models than classification and is the largest secure inference task run till date.

1 Introduction

Secure multiparty computation (MPC) allows a set of mutually distrusting parties to compute a publicly known function on their secret inputs without revealing their inputs to each other. This is done through the execution of a cryptographic protocol which guarantees that the protocol participants learn only the function output on their secret inputs and nothing else. MPC has made rapid strides - from being a theoretical concept three decades ago [13][32], to now being on the threshold of having real world impact. One of the most compelling use cases for MPC is machine learning (ML) - e.g. being able to do secure ML inference when the model and the query are private inputs belonging to different parties. There has been a flurry of recent works aimed at running inference securely with MPC such as SecureML [27], MinioNN [18], ABY3 [20], CHET [11], SecureNN [30], Gazelle [16], Delphi [19], and so on. Unfortunately, these techniques are not easy-to-use by ML developers and have only been demonstrated on small deep neural networks (DNNs) on tiny datasets such as MNIST and CIFAR. However, in order for MPC to be truly ubiquitous for secure inference tasks, it must be both easy to use by developers with no background in cryptography and capable of scaling to the DNNs used in practice.

In this work, we present CRYPFLOW, a system that converts TensorFlow inference code into MPC protocols at the push of a button. By converting code in standard TensorFlow, a ubiquitous ML framework that is used in production by various technology companies, to MPC protocols, CRYPFLOW significantly lower the entry barrier for ML practitioners and programmers to use cryptographic MPC protocols in real world applications. We make the following contributions:

First, for the developer frontend, we provide a compiler, called Athos, from TensorFlow to a variety of secure computation protocols (both 2 and 3 party) while preserving accuracy. The compiler is designed to be modular and it provides facilities for plugging in different MPC protocols. To demonstrate this modularity, we have integrated Athos with the following backends: ABY-based [12] 2-party...
computation (2PC), SCI-based 2PC \cite{25}, Aramis-based malicious secure 3-party computation \cite{17}, and Porthos-based semi-honest secure 3-party computation (3PC).

Second, for the cryptographic backend, we provide a semi-honest secure 3-party computation protocol, \textit{Porthos}, that outperforms all prior protocols for secure inference and enables us to execute, for the first time, cryptographically secure inference of ImageNet scale networks. Prior work in the area of secure inference has been limited to small networks over tiny datasets such as MNIST or CIFAR. We have evaluated \textsc{CryptFlow} on secure inference over DNNs that are at least an order of magnitude larger than the state-of-the-art \cite{5,7,8,11,16,19,21,25,27,30}. Even on MNIST/CIFAR, Porthos has lower communication complexity and is more efficient than prior works \cite{20,26,30}.

Third, we demonstrate the ease-of-use, efficiency and scalability of \textsc{CryptFlow} by evaluating on \textsc{ResNet50} \cite{14} for ImageNet classification, \textsc{DenseNet121} \cite{15} for detection of lung diseases from chest X-ray images and 3D-UNet \cite{28} for segmentation of raw 3D CT images.

Our toolchain is publicly available\footnote{https://github.com/mpc-msri/EzPC}. This paper reviews the original \textsc{CryptFlow} paper \cite{17} briefly and its increment lies in the secure segmentation evaluation (Section 5.3).

\section{Athos}

Athos compiles TensorFlow inference code to secure computation protocols. The transformations implemented in Athos are sensitive to the performance of MPC protocols. For performance reasons all efficient secure computation protocols perform computation over fixed-point arithmetic - i.e., arithmetic over integers or arithmetic with fixed precision. Athos automatically converts TensorFlow code over floating-point values into code that computes the same function over fixed-point values. This compilation is done while \textit{matching} the inference accuracy of floating-point code. Prior works \cite{16,18,21,30} in the area of running ML securely have performed this task by hand with significant losses in accuracy over floating-point code.

Athos represents a 32-bit floating-point number $r$ by a 64-bit integer $\lfloor r \cdot 2^s \rfloor$ for a precision or scale $s$. Then operations on 32-bit floating-point numbers are simulated by operations on 64-bit integers. For example $r_1 \times r_2$ is simulated as $\lfloor r_1 \cdot 2^s \rfloor \times \lfloor r_2 \cdot 2^s \rfloor$. A large $s$ causes integer overflows and a small $s$ leads to accuracy loss. To obtain a suitable scale $s$ (all variables have the same precision in Athos output), Athos works by “sweeping through” various precision levels to estimate the best precision \cite{17}.

\section{Porthos}

Porthos is an improved semi-honest 3-party secure computation protocol (tolerating one corruption) that builds upon SecureNN \cite{30}. Porthos makes two crucial modifications to SecureNN. First, SecureNN reduces convolutions to matrix multiplications and invokes the Beaver triples \cite{6} based matrix multiplication protocol. When performing a convolution with filter size $f \times f$ on a matrix of size $m \times m$, the communication is roughly $2q^2 f^2 + 2f^2 + q^2$ elements in the ring $\mathbb{Z}_{2^{64}}$, where $q = m - f + 1$. Porthos computes these Beaver triples by appropriately reshaping $m \times m$ and $f \times f$ matrices. This reduces the communication to roughly $2m^2 + 2f^2 + q^2$ ring elements. Typically the filter size, $f$, is between 1 and 11 and the communication of Porthos can be up to two orders of magnitudes less than SecureNN. Additionally, in SecureNN, the protocols for non-linear layers (such as ReLU and MaxPool) require the third party to send secret shares to the first two parties. In Porthos, we cut this communication to half by eliminating the communication of one of these shares \cite{17}. This reduces the communication in the overall ReLU and MaxPool protocols by 25%.

\section{Motivating Example}

In this section, we describe the end-to-end working of \textsc{CryptFlow} through an example of logistic regression. The toolchain is shown in Figure \ref{fig:toolchain}.

\textsc{CryptFlow} takes as input a pre-trained floating-point TensorFlow model. For example, consider the code snippet for logistic regression over MNIST dataset in TensorFlow as shown in Figure \ref{fig:code_snippet}. Our compiler first generates the TensorFlow graph dump as well as metadata containing the dimensions
5 Experiments

Overview. First, in Section 5.1 we use CRYPTFLOW for secure classification on ImageNet using the following pre-trained TensorFlow models: RESNET50 and DENSENET121. We show that the fixed-point MPC protocols generated by Athos matches the accuracy of cleartext floating-point RESNET50 and DENSENET121. We also show how the optimizations in Porthos help it outperform prior works in terms of communication complexity and overall execution time. Finally, we discuss two case-studies of running CRYPTFLOW on DNNs for medical image analysis. The compilation time of CRYPTFLOW is around 5 sec for RESNET50, 35 sec for DENSENET121 and 2 minutes for 3D UNet.

5.1 Secure Inference on ImageNet

These experiments are in a LAN setting on 3.7GHz machines, each with 4 cores and with 16 GB of RAM. The measured bandwidth between each of the machines was at most 377 MBps and the latency was sub-millisecond.

RESNET50 takes 25.9 seconds and 6.9 GB of communication; DENSENET121 takes 36 seconds and 10.5 GB of communication. We measure communication as total communication between all 3 parties.
Table 1: Accuracy of fixed- vs floating-point.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Float Top 1</th>
<th>Fixed Top 1</th>
<th>Float Top 5</th>
<th>Fixed Top 5</th>
<th>SecureNN (s)</th>
<th>Porthos (s)</th>
<th>SecureNN Comm. (GB)</th>
<th>Porthos Comm. (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESNET50</td>
<td>76.47</td>
<td>76.45</td>
<td>93.21</td>
<td>93.23</td>
<td>38.36</td>
<td>25.87</td>
<td>8.54</td>
<td>6.87</td>
</tr>
<tr>
<td>DENSENET121</td>
<td>74.25</td>
<td>74.33</td>
<td>91.88</td>
<td>91.90</td>
<td>53.99</td>
<td>36.00</td>
<td>13.53</td>
<td>10.54</td>
</tr>
</tbody>
</table>

Table 2: Porthos vs SecureNN.

- each party roughly communicates a third of this value. We show that Athos generated fixed-point code matches the accuracy of floating-code on RESNET50 and DENSENET121 in Table 1.

Detailed comparisons of CRYPTFLOW with prior works on secure inference can be found in [17]. However, since Porthos builds on SecureNN, we compare them on ImageNet scale benchmarks in Table 2. For this purpose, we add the code of SecureNN available at [3] as another backend to CRYPTFLOW. These results show that Porthos improves upon the communication of SecureNN by a factor of roughly 1.2X–1.5X and the runtime by a factor of roughly 1.4X–1.5X.

5.2 Lung diseases from 2D chest X-Ray images

In [33], the authors train a DENSENET121 to predict lung diseases from chest X-ray images. They use the publicly available NIH dataset of chest X-ray images and end up achieving an average AUROC score of 0.845 across 14 possible disease labels. These DNNs are available as pre-trained Keras models. We converted them into TensorFlow using [2] and compiled the automatically generated TensorFlow code with CRYPTFLOW. During secure inference, we observed no loss in classification accuracy and the latency is similar to the runtime of DENSENET121 for ImageNet.

5.3 Segmenting tumors and organs at risk from 3D CT images

Half a million cancer patients receive radiotherapy each year [24]. Personalized radiation treatments require segmenting tumors and organs at risk from 3D volumetric images. Currently, this segmentation is a manual process where an oncologist draws contours along regions of interest slice-by-slice across the whole volume. This process often takes several hours per image which ML provided automation [22][31] can reduce to minutes. We consider a 3D-U-Net model [28] that takes as input a raw 3D image obtained via Computed Tomography (CT) scans of the pelvic region and delineates tumor volumes and organs at risk. This model’s accuracy is within the inter-observer variability seen among clinical experts [23] and requires 1.87 Teraflops per inference.

Since this model is implemented in PyTorch, we first export it to ONNX and then use CRYPTFLOW’s ONNX frontend. For our secure inference setup, each party has 32 cores running at 2.4GHz, no GPUs, and 128GB RAM. The parties are connected on a network with ping latency 0.2s and 625MBps bandwidth. On this set up, secure inference incurs a latency of 1 hour and 57 minutes and 557GB of communication. The most expensive operators in this computation are 3D transposed convolutions (or deconvolutions) and, to the best of our knowledge, CRYPTFLOW is the only secure inference tool that supports these operations. In our experience, it takes a couple of days for a scan to reach the oncologist for review and hence this latency overhead can be acceptable.

6 Related work and conclusion

Other related systems for converting PyTorch/Tensorflow to MPC protocols [11][9][10][29] only support 3PC. Whereas, CRYPTFLOW additionally supports 2PC backends. CRYPTFLOW provides the first implementation and evaluation of a system for secure segmentation. With CRYPTFLOW, data scientists, with no background in cryptography, can obtain secure inference implementations for their trained models at the push of a button.

References


