FACULTY OF ENGINEERING TECHNOLOGY

DE NAYER (SINT-KATELIJNE-WAVER) CAMPUS

Object Detection on Encrypted Data

Using Secure Multiparty Computation

Felix LERNER

Promotor: Prof. dr. ir. Toon Goedemé
Co-promotor: Dr. Pradip Mainali (Onespan)

Masterproef ingediend tot het behalen van de graad van master of Science in de industriële wetenschappen: Industriele Wetenschappen Electronica-ICT

Academiejaar 2019 - 2020

Voorwoord

Het voorwoord vul je persoonlijk in met een appreciatie of dankbetuiging aan de mensen die je hebben bijgestaan tijdens het verwezenlijken van je masterproef en je hebben gesteund tijdens je studie.

Samenvatting

De (korte) samenvatting, toegankelijk voor een breed publiek, wordt in het Nederlands geschreven en bevat **maximum 3500 tekens**. Deze samenvatting moet ook verplicht opgeladen worden in KU Loket.

Abstract

We are affected with machine learning in many aspects of our daily lives, applications ranges from facial recognition to enhanced healthcare to self-driving cars. As companies outsource image classification tasks to cloud computing service providers, we see a rise in privacy concerns for both the users wishing to keep their data confidential, as for the company wishing to keep their classifier obfuscated.

Keywords: Computer vision, Cryptography, machine learning, secure multiparty computation, deep learning, object detetcion, privacy preserving, MLaaS, encryption

Contents

List of Figures

List of Tables

Lijst van symbolen

Maak een lijst van de gebruikte symbolen. Geef het symbool, naam en eenheid. Gebruik steeds SI-eenheden en gebruik de symbolen en namen zoals deze voorkomen in de hedendaagse literatuur en normen. De symbolen worden alfabetisch gerangschikt in opeenvolgende lijsten: kleine letters, hoofdletters, Griekse kleine letters, Griekse hoofdletters. Onderstaande tabel geeft het format dat kan ingevuld en uitgebreid worden. Wanneer het symbool een eerste maal in de tekst of in een formule wordt gebruikt, moet het symbool verklaard worden. Verwijder deze tekst wanneer je je thesis maakt.

b	Breedte	$[mm]$
A	Oppervlakte van de dwarsdoorsnede	$[mm^2]$
c	Lichtsnelheid	$[m/s]$

Lijst van afkortingen

MPC	Secure Multiparty Computation
MLaaS	Machine Learning as a Service
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit

1 Introduction

TODO: Praten over Onespan Deep learning based object detection on images is a hot topic for researchers and interest in machine learning is steadily growing among miscellaneous businesses. Secure multiparty computation (MPC) is a subfield of cryptography, making it possible for a party to run an algortihm on confidential data, that is supposed to stay unknown even to the party running the algorithm.
Both concepts aren't new concepts, but with the rise of big data and processing power, there has been an increase in research into these fields.
In this thesis we present the applicability of MPC for deep learning based object detection.

1.1 Problem

The use of third party MLaaS (Machine Learning as a Service) providers or any cloud computing solution, as processing power for an image classification task, raises privacy concerns as sensitive images of users need to be sent to servers running an instance of the neural network. It's important to note that the transport of the image from the client to the server is secure, since the parties can make use of reliable HTTPS (Hypertext Transfer Protocol Secure) connections. The user's images, however, are stored in plaintext on the server, aswell as the computed output of the image. Furthermore the whole design of the neural network including all trained parameters needs to be stored on the servers of the third party, for the image classifier to function. Both of these remote storage solutions require a considerably amount of trust in the third party. Since the third party could potentially exploit the user data for commercial purposes or even steal the intellectual property of the image classifier. In this thesis we try to tackle the need to trust a third party MLaaS provider.

We want it to compute an encrypted image on an obfuscated neural network to ouput a correct encrypted result.

1.2 Hypothesis

How can we securely compute the forward propagation of a face recognition neural network?
With the use of MPC protocols we can implement methods such that we can compute a whole neural network on an encrypted input. We predict a drastically decrease in perfomance and will try to find performance optimizations along the way of implementing a proof of concept.

2 Literature study

In this chapter we will take a quick look at how convolutional neural networks function, layer by layer. We will also learn how secure multiparty computation (MPC) in general is possible. Finaly, we will show the related work on combining these two subjects so far.

2.1 Convolutional neural network

Convolution neural networks (CNN) is a special type of neural network used for images. The spatial properties of the pixels in the image are used during the evaluation of the input, meaning the neighbouring pixels of a central pixel impact the output to the next layer of that central pixel while pixels further away do not. CNNs are made of multiple layers. Typically, as you move further from the input layer to the output layer the dimensionality reduces, we can say the input gets mapped on a desired output manifold. Inputs that we classify as similar are supposed to be in the same region in the output manifold.

2.1.1 Convolution layer

In this layer a discrete convolution of a kernel K shifting over an image I is performed, as shown in equation 2.1. The kernel has parameters also called weights, so that certain features get extracted from the input.

$$(I * K)[m,n] = \sum_{j}\sum_{k} I[m-j, n-k]K[j,k] \tag{2.1}$$

The output of this layer is a convolved feature map.

2.1.2 Activation function

Since these convolutions are simple lineair operations and most image classification tasks require non-lineair classifiers, non-lineairity needs to be added to the neural network. This is achieved through adding a non-linear activation function after a convolution or fully connected layer. The most popular activation function is the rectified linear unit (ReLU) as seen in equation 2.2.

$$f(x) = max(0,1) \tag{2.2}$$

2.1.3 Pooling layer

The dimensionality reduction we talked about in the beginning of this chapter happens in the pooling layer. A kernel is shifted over the image. In the case of max pooling the kernel selects the maximum value of the portion of the image it covers, to create the new dimensionality reduced image.

2.1.4 Fully connected layer

The fully connected layer is a multilayer perceptron that discriminates different object classes and identifies identical ones. All elements in vector h_{i-1}^{out} have their own bias B_i and weight W_i so that h_i^{in} can be calculated for each layer i according to equation 2.3.

$$h_i^{in} = h_{i-1}^{out} \cdot W_i + B_i \tag{2.3}$$

2.2 Secure multiparty computation

Secure multiparty coputation is a protocol that is used between n number of parties P. Each of these parties has private data also called a secret S. With MPC it is possible for these parties to compute a public[1] function f on the secrets. Such that a party $P_i \in P$ only knows his secret $s_i \in S$ and the public securely computed output $f(s_0, s_2, s_{n-1})$ after the protocol has succesfully finished. A classic application is Yao's Millionaires' problem Yao (1982) in wich two millionaires wish to know who is richer, there is catch however. Instead of making their balances publicly known. They wish to keep their balances a secret. In this case the number of parties n is 2 the secrets s_0 and s_1 are their balances. The public function $f(s_0, s_1) = 1$ if $s_0 < s_1$ and 0 otherwise.

We categorize 2 types of parties based on their willingness to deviate from the correct predefined protocol.

- **Honest but curious parties (passive):** Parties wish to know other parties secrets but will not deviate from the protocol at any time.
- **Malicious parties (active):** Parties wish to know other parties secrets and wish to change output of computation to favourable result. Parties will deviate from the protocol to cheat and change the outcome at any time.

[1]public or global means known to all parties, while private or local means known only by the corresponding party

If the two millionaires are honest but curious parties, they will not deviate from the protocol and they will computer the correct output as a result they will know who is the richer millionair but they won't know how much money the other one has. In the other case one of the two millionaires is corrupt and acts maliciously, the honest millionair will follow protocol while the dishonest millionair will deviate from the protocol to change the result in his favour. In the event that the dishonest millionair is poorer he will change the outcome thus appearing richer. From now on, we assume the parties are honest but curious parties, unless specified otherwise. We also assume the communication between the different parties to be secure.

2.2.1 Secret sharing

In order to do secure computing, the parties need to split their secret into secret shares. A secret sharing method can be used by the secret holder to split a secret into a number of shares. Combining these shares will reveal the secret, while individual shares alone will leak nothing about the secret. In a (t,n) threshold secret sharing sheme parties must combine atleast t shares of the total n shares, to obtain the secret. We can now set a threshold t high enough, denying the secret to small curious parties and allowing to reveal the secret when a majority ($\geq t$) consensus is reached. Shamir's secret sharing scheme Shamir (1979) is based on polynomial interpolation and the essential idea is that it takes atleast t points in order to define a polynomial $p(x)$ of degree $t-1$. Given a set of t points in a 2-dimensional carthesian system $(x_1,y_1),(x_2,y_2),...,(x_t,y_t)$, there exists only one polynomial of degree $t-1$. This can be proven and the mathematical construction of a polynomial $p(x)$ of degree $t-1$ based on a set of t points can be calculated using Lagrange's interpolation formula 2.4.

$$p(x) = \sum_{i=1}^{t} y_i \delta_i(x) \quad \text{with} \quad \delta_i(x) = \prod_{1 \leq j < t; i \neq j} \frac{x - x_i}{x_j - x_i} \tag{2.4}$$

With this in mind, a secret dealer can now share his secret s to n parties by choosing a random $t-1$ degree polynomial $p(x) = a_0 + a_1 x + ... + a_{t-1} x^{t-1}$ in which a_0 is the secret or the number representation of the secret if the secret is not a number. The dealer now calculates n points on the polynomial starting from $x = 1$, because the secret is located at $x = 0$. Each party $P_i \in P_1, P_2, ..., P_n$ is given a different single point (x_i, y_i), at this stage the secret is shared. The convention for the notation of a shared secret is $[s]$. We can say a party P_i is holding a share in the form of a point (x_i, y_i) or shortened $[s]_i$.

To recombine the secret, the parties simply broadcast or send their shares to a central entity, if more than t shares are known, it suffices to calculate the Lagrange polynomial $p(x)$ and $s = p(0)$. In the case of not having enough shares, the Lagrange polynomial containing the secret becomes impossible to calculate since every polynomial is equally likely, thus revealing absolutely nothing about the secret.

Note that all arithmetic in this section can be done over some finite field $\mathbb{F}_q$ to speed up the algorithms.

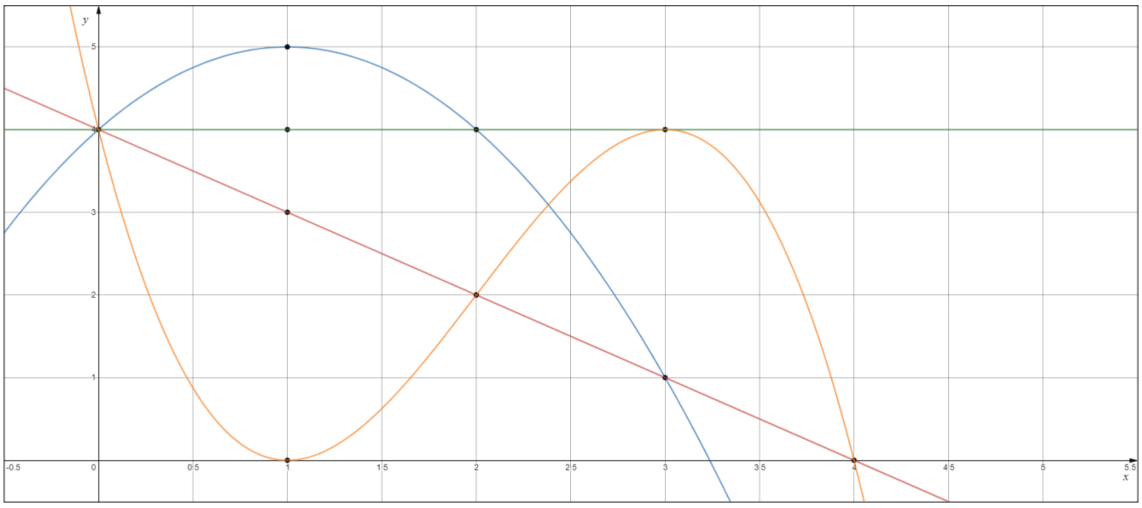

Figure 2.1: Shares for different t with secret $s = 4$

A Generalisation of this (t,n)-Shamir secret sharing scheme from thesis de Hoogh (2012) as follows (Protocol 2.1 and Protocol 2.2):

1. **Share Generation:** To share $s \in \mathbb{F}_q$, the dealer generates random $a_1,...,a_t \in \mathbb{F}_q$ and puts $p(x) = s + a_1x + ... + a_tx_t$. Then the dealer computes $[s]_i = p(i)$.
2. **Share Distribution:** For each $i \in 1,...,n$, the dealer sends $[s]_i$ to party P_i.
3. **Secret Reconstruction:** Let $D \subset 1,...,n$ be a set of size $t+1$. Each party P_i for $i \in D$ sends his share $[s]_i$ to all parties. Then, each party reconstructs the secret via Lagrange interpolation.

In figure 2.1 a visualisation of this scheme shows us that in order to get the secret value, $s = 4$ in this case, we need to know atleast $t+1$ points for a given t. The trivial case where $t = 0$ (green line) is simple to understand. The secret is simply the y-value of the given point. Of course this is a rather useless approach to secret sharing since you are giving away the cleartext secret. The full potential of secret sharing becomes true for cases where $t > 0$. Two points are needed for a first order polynomial (red line), three points for a second order polynomial and so on.
It's important to make sure the parties are distributed and to try minimizing the incentive to collude. Distribution is needed to lower the risk of having a malicious party taking over control of the network, this can be done by splitting the parties up to many small stakeholders instead of a couple centralized stakeholders. We can only totally avoid collusion if the secret holder partakes to the MPC.

2.2.2 Operations

A neural network can be seen as an enormous function with millions of coefficients. Lucky for us the function can be broken up into 3 different operations: addition and multiplication in fully connected and convolution layers and a relational operator for max pooling and ReLU activation function. We will now show how to securely compute each of these basic operations.

2.2.2.1 Arithmetic operators

Since Shamir's secrete sharing scheme is a linear sharing scheme, each party Pi can locally compute any linear combination of a public or secret value with their secret share $[s]_i$. This gives us following operations:

- **Addition of secret and public value ($[c] \leftarrow [a] + \beta$):** Each party P_i locally adds the public value $\beta \in \mathbb{F}_q$ to it's share $[a]_i$, resulting in the new share $[c]_i = [a]_i + \beta$. Since all parties add the same β, this value is a constant.
- **Multiplication of secret and public value ($[c] \leftarrow [a] \cdot \beta$):** Each party P_i locally multiplies the public (constant) value $\beta \in \mathbb{F}_q$ with it's share $[a]_i$, resulting in the new share $[c]_i = [a]_i \cdot \beta$.
- **Addition of multiple secrets ($[c] \leftarrow [a] \cdot [b]$):** Each party P_i locally adds it's secret shares $[a]_i$ and $[b]_i$, resulting in the new share $[c]_i = [a]_i + [b]_i$.

The last operation is visually demonstrated in figure 2.2. In this example secret $a = 1$ (green) and secret $b = 3$ (red). After each party P_i locally computes the addition of $[a]_i$ and $[b]_i$ and stores it as a new share $[c]_i$. After broadcasting the new, computated share $[c]$. They can recombine the shares via Lagrange interpolation to get the polynomial of the combined shares $[c]$ and more importantly the output of the addition of the two secrets a and b. This happens all with zero knowledge about a or b. Since the computations are done locally no communication other than the inital share distribution is needed, the round complexity is 1.

The inverse operations (subtraction and division) are possible too, and the protocol is the same.
Multiplication of two secret values is more challenging since it isn't a linear operation. Their are multiple methods to perform secret multiplication, the method described in Ben-Or et al. (1988) also called the BGW protocol (initials of the authors) has to solve two problems along the way of computing $c = a \cdot b$. Assume a and b are respectively encoded by $f(x)$ and $g(x)$ and $n \geq 2t + 1$. The free coefficient of $h(x) = f(x) \cdot g(x)$ is simply $h(0) = f(0) \cdot g(0)$, this means a simple multiplication of the polynomials would be sufficient to compute the multiplication of the secrets. There is however a major problem, multiplication of two polynomials of degree t yields a polynomial of degree $2t$.

While this poses no problem with interpolating $h(x)$ from it's shares $[c]$ since $n \geq 2t + 1$, further multiplications will continue to raise the degree to a level where $t > n$ making it impossible to interpolate the resulting polynomial $h(x)$. The second problem is harder to spot. The polynomial $h(x)$ as a result of the multiplication of polynomials $f(x)$ and $g(x)$ is reducible (since it's a multiplication).

In other words $f(x)$ and $g(x)$ are uniformly random polynomials of degree t, but $h(x)$ as a multiplication of two random polynomials is not irreducible, therefore $h(x)$ is not uniformly random. To make sure the resulting polynomial stays of degree t and is uniformly random, the parties run a protocol to generate a random polynomial of degree $2t$. This protocol works as follows. Each party

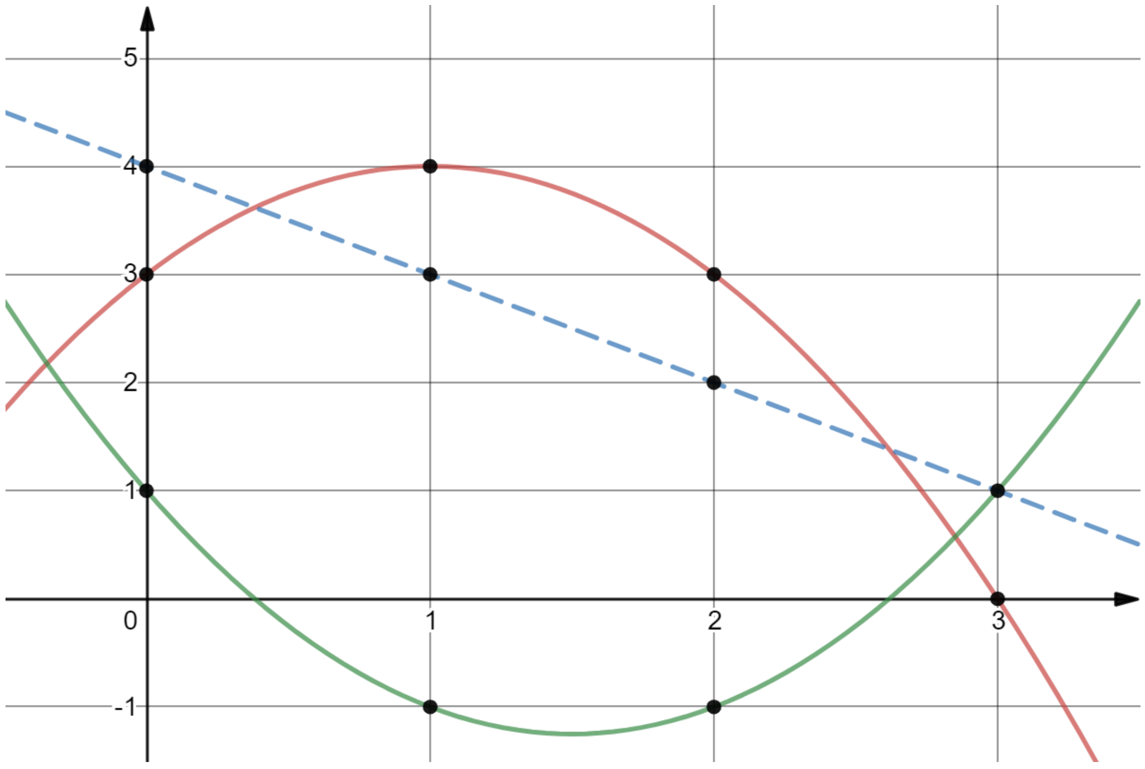

Figure 2.2: Adding two secrets for $n = 3$ parties

P_i randomly selects a private polynomial $q_i(x)$ with secret $q_i(0) = 0$ of degree $2t$ and distributes it's shares among the parties. Each party P_i now has n random shares $[q]_i^k$ (with $k : 1 \to n$). After each party adds all it's random shares they hold a secret, truly random polynomial $q(x)$ with a zero as free coefficient $q(0) = 0$.

Each party P_i now computes the multiplication of the two secret values $[a]$ and $[b]$ and instead of using $[c]$ encoded as $h(x)$ we can add the random polynomial to the result, thus randomizing the coefficients and making the polynomial uniformly random. This will not mess up the result as now $a \cdot b$ is $(f(0) \cdot g(0)) + q(0)$ and $q(0)$ equals zero. This step is also called the randomization step.

The parties now run a protocol to reduce the degree of $h(x)$ to t. This protocol can be computed locally (no communication is required) by multiplying the shares of the polynomial $h(x)$ to a specific, matrix of constants. This will truncate the result $h(x)$ of degree $2t$ to a polynomial of degree t. Proof for this degree reduction step can be found in the study by Asharov and Lindell (2017).

After these two steps the parties now hold $c = a \cdot b$ encoded by $h(x)$ of degree t with coefficients uniformly distributed. They can now recombine their shares to find the polynomial $h(x)$ and the product of a and b, $h(0)$.

2.2.2.2 Relational operators

2.3 Related work

2.4 Conclusion

3 Implementation

3.1 Specifications

3.2 Design

3.3 Conclusion

4 Evaluation

4.1 Results

4.1.1 Reliability results

4.1.2 Timing results

4.2 Discussion

4.3 Conclusion

5
Conclusion

Bibliography

Asharov, G. and Lindell, Y. (2017). A full proof of the bgw protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151.

Ben-Or, M., Goldwasser, S., and Wigderson, A. (1988). Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM.

de Hoogh, S. J. A. (2012). Design of large scale applications of secure multiparty computation: secure linear programming.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.

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

A
Uitleg over de appendices

Bijlagen worden bij voorkeur enkel elektronisch ter beschikking gesteld. Indien essentieel kunnen in overleg met de promotor bijlagen in de scriptie opgenomen worden of als apart boekdeel voorzien worden.

Er wordt wel steeds een lijst met vermelding van alle bijlagen opgenomen in de scriptie. Bijlagen worden genummerd het een drukletter A, B, C,...

Voorbeelden van bijlagen:

Bijlage A:	Detailtekeningen van de proefopstelling
Bijlage B:	Meetgegevens (op USB)

FACULTY OF ENGINEERING TECHNOLOGY
DE NAYER (SINT-KATELIJNE-WAVER) CAMPUS
Jan De Nayerlaan 5
2860 SINT-KATELIJNE-WAVER, België
tel. + 32 16 30 10 30
fet.denayer@kuleuven.be
www.fet.kuleuven.be