

# 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: Industriële  
Wetenschappen Electronica-ICT

Academiejaar 2019 - 2020

©Copyright KU Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor(en) als de auteur(s) is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, kan u zich richten tot KU Leuven Technologicampus De Nayer, Jan De Nayerlaan 5, B-2860 Sint-Katelijne-Waver, +32 15 31 69 44 of via e-mail [iiw.denayer@kuleuven.be](mailto:iiw.denayer@kuleuven.be).

Voorafgaande schriftelijke toestemming van de promotor(en) is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

# 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

<b>Voorwoord</b>	<b>iii</b>
<b>Samenvatting</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>Inhoud</b>	<b>vii</b>
<b>Figurenlijst</b>	<b>viii</b>
<b>Tabellenlijst</b>	<b>ix</b>
<b>Symbolenlijst</b>	<b>x</b>
<b>Lijst met afkortingen</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem . . . . .	1
1.2 Hypothesis . . . . .	2
<b>2 Literature study</b>	<b>3</b>
2.1 Convolutional neural network . . . . .	3
2.1.1 Convolution layer . . . . .	3
2.1.2 Activation function . . . . .	4
2.1.3 Pooling layer . . . . .	4
2.1.4 Fully connected layer . . . . .	4
2.2 Secure multiparty computation . . . . .	4
2.2.1 Secret sharing . . . . .	5
2.2.2 Operations . . . . .	6
2.3 Related work . . . . .	7
2.4 Conclusion . . . . .	7

---

<b>3</b>	<b>Implementation</b>	<b>9</b>
3.1	Specifications . . . . .	9
3.2	Design . . . . .	9
3.3	Conclusion . . . . .	9
<b>4</b>	<b>Evaluation</b>	<b>10</b>
4.1	Results . . . . .	10
4.1.1	Reliability results . . . . .	10
4.1.2	Timing results . . . . .	10
4.2	Discussion . . . . .	10
4.3	Conclusion . . . . .	10
<b>5</b>	<b>Conclusion</b>	<b>11</b>
<b>A</b>	<b>Uitleg over de appendices</b>	<b>13</b>

## List of Figures

2.1	Shares for different $t$ with secret $s = 4$ . . . . .	6
2.2	Adding two secrets for $n = 3$ parties . . . . .	8



## 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 algorithm 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 output 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 performance 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. Finally, 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.

$$(I * K)[m, n] = \sum_j \sum_k I[m - j, n - k] K[j, k] \quad (2.1)$$

The output of this layer is a convolved feature map.

### 2.1.2 Activation function

Since these convolutions are simple linear operations and most image classification tasks require non-linear classifiers, non-linearity 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.

$$f(x) = \max(0, 1) \quad (2.2)$$

### 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.

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

## 2.2 Secure multiparty computation

Secure multiparty computation 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<sup>1</sup> 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 successfully finished. A classic application is Yao's Millionaires' problem Yao (1982) in which 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.

<sup>1</sup>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 compute the correct output as a result they will know who is the richer millionaire 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 millionaire will follow protocol while the dishonest millionaire will deviate from the protocol to change the result in his favour. In the event that the dishonest millionaire 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 scheme 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 cartesian system  $(x_1, y_1), (x_2, y_2), \dots, (x_t, y_t)$ , 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.

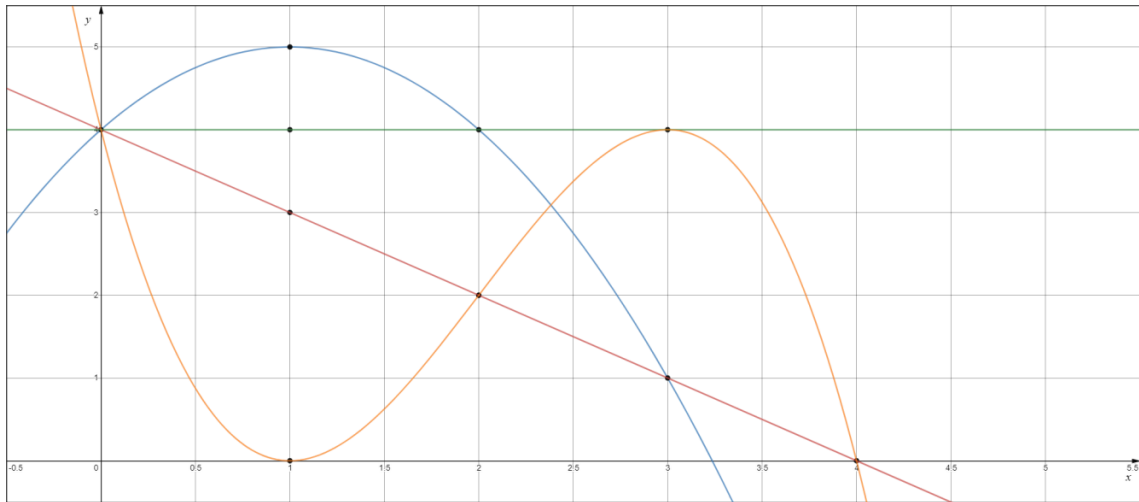
$$p(x) = \sum_{i=1}^t y_i \delta_i(x) \quad \text{with} \quad \delta_i(x) = \prod_{1 \leq j < t; j \neq i} \frac{x - x_j}{x_i - x_j} \quad (2.4)$$

With this in mind, a secret dealer can now share his secret  $s$  to  $n$  parties by choosing a random  $t - 1$  degree polynomial  $p(x) = a_0 + a_1x + \dots + a_{t-1}x^{t-1}$  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  $P_i \in P_1, P_2, \dots, P_n$  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  $a_1, \dots, a_t \in \mathbb{F}_q$  and puts  $p(x) = s + a_1x + \dots + a_tx_t$ . Then the dealer computes  $[s]_i = p(i)$ .
2. **Share Distribution:** For each  $i \in 1, \dots, n$ , the dealer sends  $[s]_i$  to party  $P_i$ .
3. **Secret Reconstruction:** Let  $D \subset 1, \dots, n$  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 secret sharing scheme is a linear sharing scheme, each party  $P_i$  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 its share  $[a]_i$ , resulting in the new share  $[c]_i = [a]_i + \beta$ . Since all parties add the same  $\beta$ , 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 its share  $[a]_i$ , resulting in the new share  $[c]_i = [a]_i \cdot \beta$ .
- **Addition of multiple secrets** ( $[c] \leftarrow [a] + [b]$ ): Each party  $P_i$  locally adds its 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, computed share  $[c]$ . They can recombine the shares via Lagrange interpolation to get the output of the addition of the two secrets  $a$  and  $b$ . This happens all with zero knowledge about  $a$  or  $b$ .

#### Addition

#### Multiplication

### 2.2.2.2 Relational operators

## 2.3 Related work

## 2.4 Conclusion

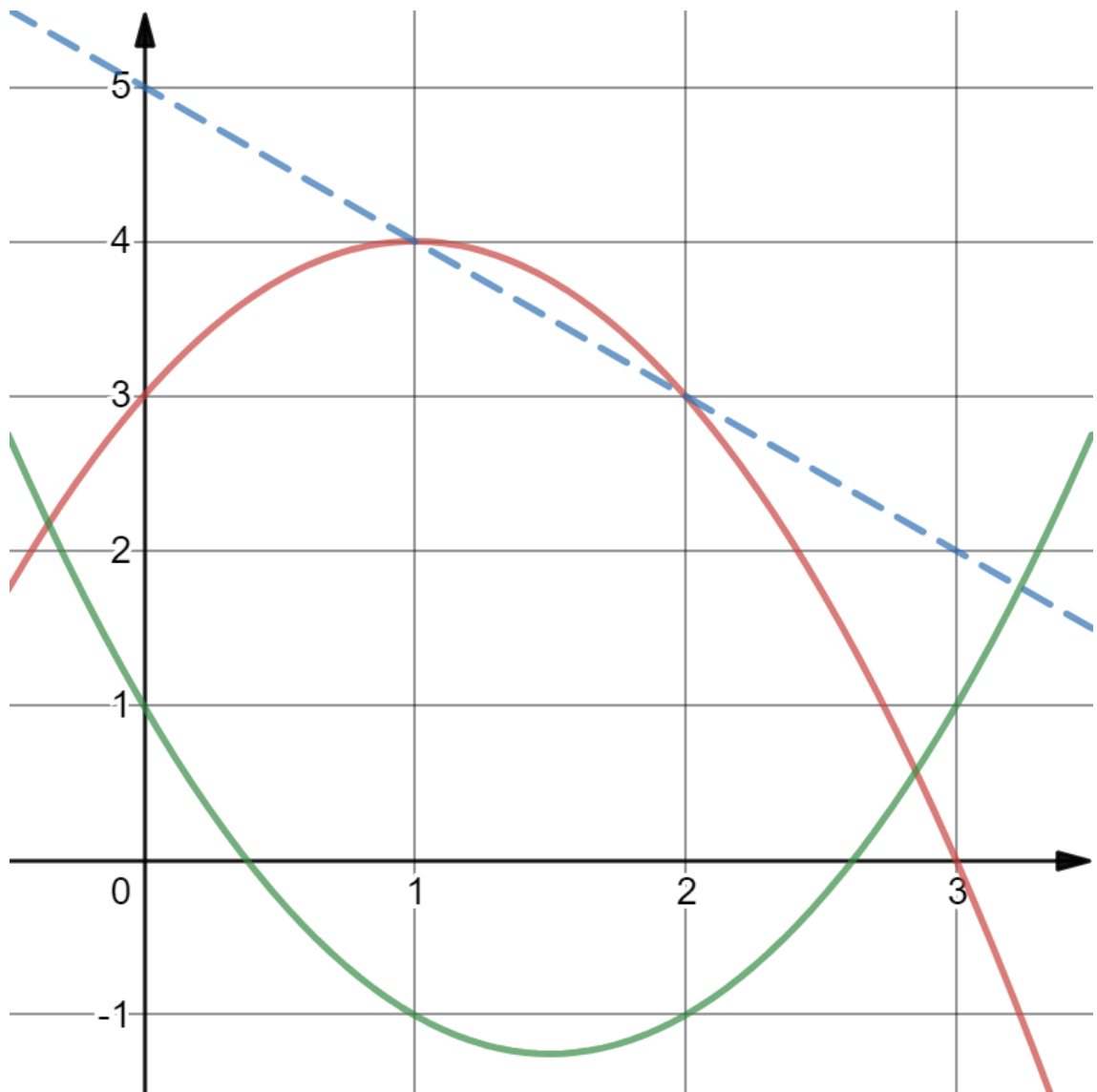


Figure 2.2: Adding two secrets for  $n = 3$  parties

# 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

- 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.



## 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 met 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](http://www.fet.kuleuven.be)

