

Peter Julius Waldert

## Secure Classification as a Service

### BACHELOR'S THESIS

Bachelor's degree programmes:  
*Physics and Information & Computer Engineering*

#### Supervisors

Dipl.-Ing. Roman Walch

Dipl.-Ing. Daniel Kales

Institute of Applied Information Processing and Communications  
Graz University of Technology

# Abstract

Abstract of your thesis (at most one page)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

**Keywords:** FHE, classification, neural network

**Technologies:** Microsoft SEAL (C++, nodejs), Tensorflow Keras, Numpy, xtensor, Docker, msgpack, React, Materialize, Nginx

**Languages:** C++, Python, JavaScript

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Background</b>	<b>6</b>
2.1	Basics of Fully Homomorphic Encryption . . . . .	6
2.1.1	Packing . . . . .	6
2.1.2	HE using RSA . . . . .	6
2.1.3	Learning with Errors (LWE) . . . . .	7
2.1.4	Ring-LWE . . . . .	7
2.1.5	The BFV scheme . . . . .	7
2.1.6	The CKKS scheme . . . . .	7
2.2	Machine Learning . . . . .	8
2.2.1	Linear Regression . . . . .	8
2.2.2	Gradient Descent . . . . .	8
2.2.3	Multi-Layered Neural Networks . . . . .	8
2.2.4	The Backpropagation Algorithm . . . . .	9
2.3	Post-Quantum Security . . . . .	9
2.3.1	Shor's Algorithm . . . . .	9
2.4	Demo . . . . .	9
2.5	Notation and Acronyms . . . . .	9
<b>3</b>	<b>Implementation</b>	<b>10</b>
3.1	Chosen Software Architecture . . . . .	10
3.1.1	Docker Multi-Stage Build . . . . .	10
3.2	The MNIST dataset . . . . .	10
3.3	Matrix-Vector Multiplication . . . . .	10
3.3.1	Adapting to non-square matrices . . . . .	11
3.3.2	The Naïve Method . . . . .	11
3.3.3	The Diagonal Method . . . . .	12
3.3.4	The Hybrid Method . . . . .	12
3.3.5	The Babystep-Giantstep Optimization . . . . .	12
3.4	Polynomial Evaluation . . . . .	14
3.5	Transparent Ciphertext . . . . .	15
<b>4</b>	<b>Results</b>	<b>16</b>
4.1	Performance / Runtime Benchmarks . . . . .	16
4.2	Communication Overhead . . . . .	16
4.3	Accuracy . . . . .	16

5 Conclusion	17
Notation	18
Acronyms	19
Bibliography	20

# Chapter 1

## Introduction

Goal:

# Chapter 2

## Background

### 2.1 Basics of Fully Homomorphic Encryption

**Homomorphic Encryption (HE)** makes it possible to operate on data without knowing it. One can distinguish three flavors of it, Partial-, Semi- and **Fully Homomorphic Encryption (FHE)**.

- Brakerski/Fan-Vercauteren (BFV) scheme for integer arithmetic
- Brakerski-Gentry-Vaikuntanathan (BGV) scheme for integer arithmetic
- Cheon-Kim-Kim-Song (CKKS) scheme for real-number arithmetic
- Ducas-Micciancio (FHEW) and Chillotti-Gama-Georgieva-Izabachene (TFHE) schemes for Boolean circuit evaluation

#### 2.1.1 Packing

FFT in CKKS! Polynom  $\rightarrow$  Vektor mit einer FFT Vektor  $\rightarrow$  Polynom mit einer IFFT

#### 2.1.2 HE using RSA

With unpadded RSA, some arithmetic can be performed on the ciphertext - looking at the encrypted ciphertext  $\mathcal{E}(m_1) = (m_1)^r \bmod n$  of the message  $m_1$  and  $m_2$  respectively, the following holds:

$$\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) = (m_1)^r (m_2)^r \bmod n \quad (2.1)$$

$$= (m_1 m_2)^r \bmod n \quad (2.2)$$

$$= \mathcal{E}(m_1 \cdot m_2) \quad (2.3)$$

The encryption therefore partially fulfills the properties of ring homomorphism, which in general is defined as follows:

#### Definition 2.1.1: Ring Homomorphism

Given two rings  $(R, +, \cdot)$  and  $(S, \oplus, \otimes)$ , we call a mapping  $\varphi : R \rightarrow S$  a ring homomorphism when it satisfies the following conditions:

$$\forall a, b \in R : \varphi(a + b) = \varphi(a) \oplus \varphi(b) \wedge \varphi(a \cdot b) = \varphi(a) \otimes \varphi(b)$$

### **2.1.3 Learning with Errors (LWE)**

### **2.1.4 Ring-LWE**

Learning with Errors on Rings (RLWE)

how to get from LWE to RLWE

### **2.1.5 The BFV scheme**

### **2.1.6 The CKKS scheme**

The CKKS scheme allows us to perform approximate arithmetic on floating point numbers.

## 2.2 Machine Learning

Undoubtedly one of the most prevalent concepts in today's computing world, **Machine Learning (ML)** has shaped how computers think and how we interact with them significantly. As Shafi GOLDWASSER puts it, 'Machine Learning is somewhere in the intersection of Artificial Intelligence, Statistics and Theoretical Computer Science' (Goldwasser 2018).

Within the scope of this thesis, the basics of neural networks and associated learning methods shall be covered, limited to the category of supervised learning problems (as opposed to unsupervised learning problems). Supervised learning refers to the machine *training* an algorithm to match some input data (features) with corresponding output data (targets), often related to pattern recognition. The trained algorithm can then be utilised to match fresh input data with a prediction of the targets.

A popular subset of applications to **ML** are classification problems, predominantly image classification, which was not as easily possible before without a human eye due to the lack of computing power. Classification problems can be formulated quickly, the goal is to computationally categorize input data (for instance, images) into a predefined set of classes (for instance, cats and dogs). The primary concept behind **Machine Learning** is not at all new, linear regression was already employed by GAUSS and LEGENDRE in the early 19<sup>th</sup> century; the term 'Neural Network' was first used by MCCULLOCH and PITTS in 1943. Much media attention was earned in the 2000-2010 decade when larger image classification problems became feasible with the increasing computational power of modern computers, up until the advent of Deep Learning (Bishop and Nasrabadi 2007).

### 2.2.1 Linear Regression

Given an input vector  $\mathbf{x} \in \mathbb{R}^n$ , the goal of linear regression is to predict the value of a target  $t \in \mathbb{R}$ , according to some model  $M$ .

### 2.2.2 Gradient Descent

### 2.2.3 Multi-Layered Neural Networks

#### Theorem 2.2.1: Universal Approximation Theorem

If the neural network has at least one hidden layer, proper nonlinear activation functions and enough data and hidden units, it can approximate any continuous function  $y(x, w) : \mathbb{R}^n \mapsto \mathbb{R}$  arbitrarily well on a compact domain (Hornik, Stinchcombe and White 1989).

Matrix -> Activation Function

- Matrix Multiplication (Dense Layer)
- Convolutional Layer
- Sigmoid Activation
- Max Pooling



### 2.2.4 The Backpropagation Algorithm

## 2.3 Post-Quantum Security

### 2.3.1 Shor's Algorithm

## 2.4 Demo

In this chapter, we provide some usage examples for glossaries and acronym lists with `glossaries` (section 2.5), bibliography and citations with `biblatex` (section 2.5), and more.

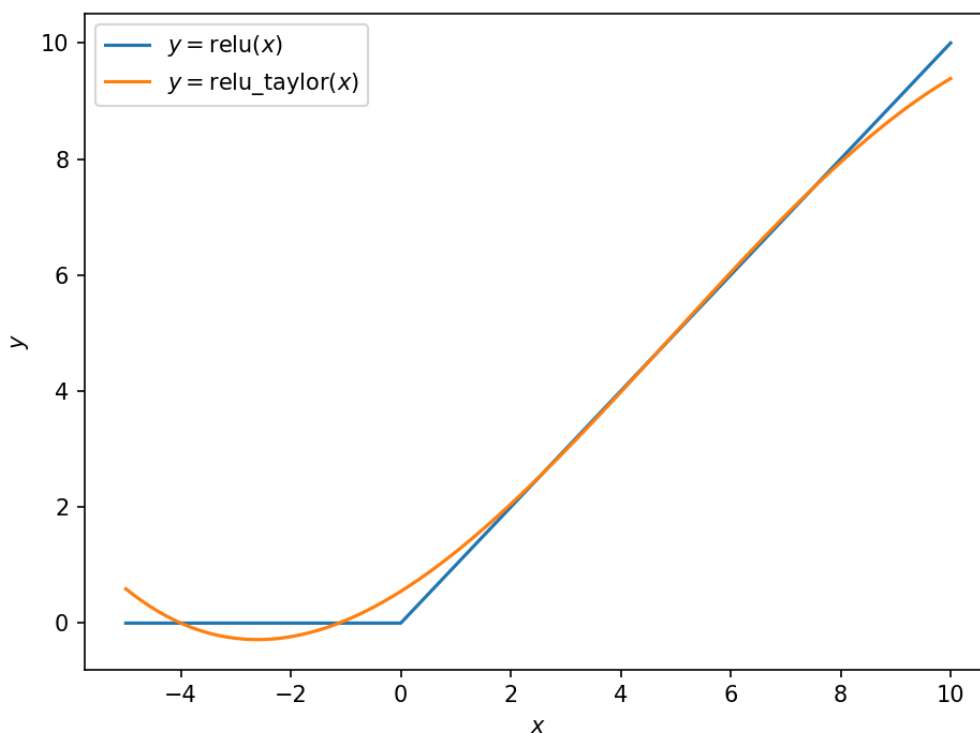


Figure 2.1: Comparison of the Relu activation function vs. its Taylor expansion

## 2.5 Notation and Acronyms

Symbols and acronyms are defined in the preamble, after loading the `glossaries` package, and used as follows.

In this chapter, we introduce the necessary background on the `Advanced Encryption Standard (AES)`. We denote binary exclusive-or by  $\oplus$ .

This is an example of how to specify and cite a book `Daemen and Rijmen 2002`, a journal article `Shannon 1949`. `AES` is a block cipher defined by `Daemen and Rijmen (2002)`.

# Chapter 3

## Implementation

### 3.1 Chosen Software Architecture

In the given setting, the most accessible frontend is commonly a JavaScript web application.

To still make the classification run as quickly and efficiently as possible, a C++ binary runs in the backend providing an HTTP API to the frontend application. In order to allow for more flexibility of the HTTP server, the initial approach was to pipe requests through a dedicated web application framework with database access that would allow, for instance, user management next to the basic classification. However, the resulting communication and computation overhead, even when running with very efficient protocols such as ZeroMQ, was too high.

Extending the accessibility argument to reproducibility, Docker is a very solid choice (Nüst et al. 2020). To run the attached demo project, simply execute

```
1  docker-compose build
2  docker-compose up
```

in the 'code' folder and point your browser to <https://localhost>.

#### 3.1.1 Docker Multi-Stage Build

An enterprise-grade, scalable deployment is achieved by means of zero-dependency Alpine Linux images which contain nothing but compiled binaries and linked libraries.

### 3.2 The MNIST dataset

The MNIST dataset (LeCun and Cortes 1998) contains X train and Y test images with corresponding labels. In order to stick to the traditional feedforward technique with data represented in vector format, therefore it is common to reshape data from (28, 28) images (represented as grayscale values in a matrix) into a 784 element vector.

### 3.3 Matrix-Vector Multiplication

The dot product that is required as part of the neural network evaluation process needs to be implemented on SEAL ciphertexts as well.

There are multiple methods to achieve a syntactically correct dot product (matrix-vector multiplication) as described by [Juvekar, Vaikuntanathan and Chandrakasan \(2018\)](#) for (square) matrices.

1. **Naïve MatMul** - very simple to derive but impractical in practice due to the limited further applicability of the result consisting of multiple ciphertexts. Applicable to arbitrary matrix dimensions, i.e. matrices  $M \in \mathbb{R}^{s \times t}$ , of course limited by the unreasonably high memory consumption and computation time of this approach.
2. **Diagonal MatMul** - a simple and practical solution applicable to square matrices  $M \in \mathbb{R}^{t \times t}$  that has a major advantage compared to the previous method as the computation yields a single ciphertext object instead of many which can be directly passed on to a following evaluation operation.
3. **Hybrid MatMul** - essentially extending the diagonal method by generalising the definition of the diagonal extraction mechanism to 'wrap around' in order to match the dimensionality of the input vector. Applicable to arbitrary matrix dimensions, i.e. matrices  $M \in \mathbb{R}^{s \times t}$  and favourable compared to the Naïve Method.
4. **Babystep-Giantstep MatMul** - a more sophisticated technique aiming to significantly reduce the number of Galois rotations as they are rather expensive to carry out, with a performance boost especially noticeable for higher matrix dimensions.

For the following, define

$$\begin{aligned} \text{rot}_j : \mathbb{R}^t &\mapsto \mathbb{R}^t, \{\text{rot}_j(\mathbf{x})\}_i = x_{i+j} \\ \text{diag}_j : \mathbb{R}^{t \times t} &\mapsto \mathbb{R}^t, \{\text{diag}_j(M)\}_i = M_{i,(i+j)} \end{aligned}$$

with all indices  $i, j \in \mathbb{Z}_t$  member of the cyclic quotient group  $\mathbb{Z}_t := \mathbb{Z}/t\mathbb{Z}$  of all integers modulo  $t$ , meaning that overflowing indices simply wrap around again starting at index 0 to simplify notation. For the sake of compactness, we stick to this notation for the rest of this section.

### 3.3.1 Adapting to non-square matrices

The weight matrices in the given classification setting are by no means square, on the contrary their output dimension tends to be much lower than the input dimension as the goal is to reduce it from  $28^2 = 784$  to 10 overall.

However, that also means one cannot directly apply the diagonal method as described in the proceedings above. This 'flaw' can be mitigated by a simple zero-padding approach in order to make the matrix square, filling in zeroes until the lower dimension reaches the higher one.

### 3.3.2 The Naïve Method

Term by term, one can express a matrix-vector product of  $M \in \mathbb{R}^{s \times t}$  and  $\mathbf{x} \in \mathbb{R}^s$  as follows:

$$\{M\mathbf{x}\}_i = \sum_{j=1}^t M_{ij}x_j$$

Accordingly, a natural (or rather, naïve) way to model this multiplication in *Microsoft SEAL* would be to

1. encode each  $i$ -th matrix row  $(M_{i,1}, M_{i,2}, \dots, M_{i,t})$  using the **Encoder** with matching parameters to the ciphertext of the encoded vector  $\mathbf{x}$ .

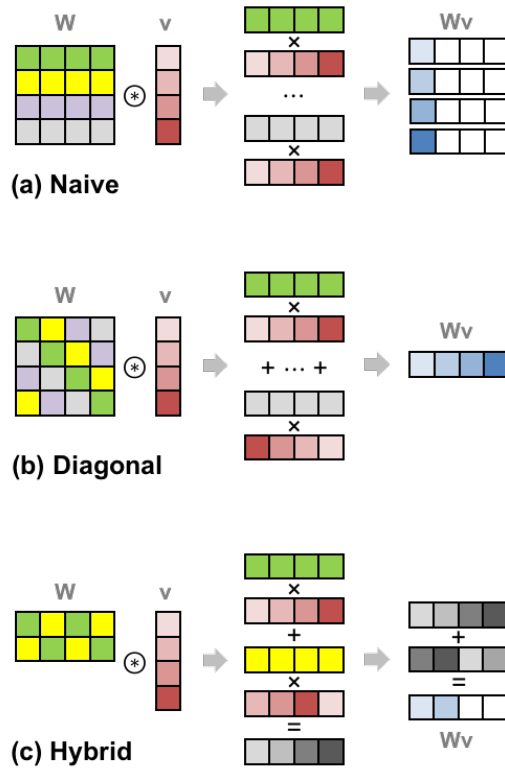


Figure 3.1: Different techniques to compute a dot product between a matrix and a vector, each having their up- and downsides.

2. multiply each encoded row with the encrypted vector using `Evaluator.multiply_plain()` to obtain the ciphertext vector  $\mathbf{y}_i \in \mathbb{R}^s$  for row  $i$ .
3. perform the 'rotate-and-sum' algorithm (Juvekar, Vaikuntanathan and Chandrakasan 2018) on each resulting vector (ciphertext)  $\mathbf{y}_i$  to obtain the actual dot product of the matrix row with the vector  $\mathbf{x}$ :
  - (a) using Galois automorphisms, rotate the entries of  $\mathbf{y}_i$  by  $\frac{s}{2}$  elements to obtain  $\text{rot}_{\frac{s}{2}}(\mathbf{y}_i)$ .
  - (b) perform an element-wise sum  $\mathbf{y}_i + \text{rot}_{\frac{s}{2}}(\mathbf{y}_i)$  whose first (and also second) half now contains the sum of the two halves of  $\mathbf{y}_i$ .
  - (c) repeat the previous two steps  $\log_2(s)$  times, halving the split parameter  $s$  each time until one obtains 1 element, which yields us the requested sum of all entries  $\sum_{k=1}^s \{\mathbf{y}_i\}_k$  as the dot product of  $\mathbf{x}$  and  $\mathbf{y}_i$ .
4. Given all the 'scalar' results of each row-vector dot product, we can construct the resulting matrix-vector product.

### 3.3.3 The Diagonal Method

### 3.3.4 The Hybrid Method

### 3.3.5 The Babystep-Giantstep Optimization

Since Galois rotations are the most computationally intensive operations in most cryptographic schemes used today (Dobraunig et al. 2021), they take a large toll on the efficiency. In order

to reduce the number of rotations required, one can make use of the *Babystep-Giantstep* optimization as described in Halevi and Shoup 2018, which works as follows:

**Theorem 3.3.1: Babystep-Giantstep Optimization**

Given a matrix  $M \in \mathbb{R}^{t \times t}$  and a vector  $\mathbf{x} \in \mathbb{R}^t$ , with  $t = t_1 \cdot t_2$  split into two BSGS parameters  $t_1, t_2 \in \mathbb{N}$  and

$$\text{diag}'_p(M) = \text{rot}_{-\lfloor p/t_1 \rfloor \cdot t_1}(\text{diag}_p(M))$$

, one can express a matrix-vector multiplication as follows:

$$M\mathbf{x} = \sum_{k=0}^{t_2-1} \text{rot}_{(kt_1)} \left( \sum_{j=0}^{t_1-1} \text{diag}'_{(kt_1+j)}(M) \cdot \text{rot}_j(\mathbf{x}) \right) \quad (3.1)$$

where  $\cdot$  denotes an element-wise multiplication of two vectors.

*Proof.* Starting from the adapted matrix-multiplication expression  $P = (P_1, P_2, \dots, P_t)^T \in \mathbb{R}^t$ , we want to show that we indeed end up with an authentic matrix-vector product.

$$P = \left\{ \sum_{k=0}^{t_2-1} \text{rot}_{(kt_1)} \left( \sum_{j=0}^{t_1-1} \text{diag}'_{(kt_1+j)}(M) \cdot \text{rot}_j(\mathbf{x}) \right) \right\}_i = \sum_{k=0}^{t_2-1} \sum_{j=0}^{t_1-1} m'_{kt_1+j, (i+kt_1)} x_{(i+kt_1)+j}$$

with

$$m'_{p,i} = \left\{ \text{diag}'_p(M) \right\}_i = \left\{ \text{rot}_{-\lfloor p/t_1 \rfloor \cdot t_1}(\text{diag}_p(M)) \right\}_i = M_{i - \lfloor \frac{p}{t_1} \rfloor t_1, i - \lfloor \frac{p}{t_1} \rfloor t_1 + p}$$

and therefore

$$\begin{aligned} m'_{kt_1+j,i} &= M_{i - \lfloor \frac{kt_1+j}{t_1} \rfloor t_1, i - \lfloor \frac{kt_1+j}{t_1} \rfloor t_1 + kt_1 + j} \\ &= M_{i - kt_1 - \lfloor \frac{j}{t_1} \rfloor t_1, i - kt_1 - \lfloor \frac{j}{t_1} \rfloor t_1 + kt_1 + j} \\ &= M_{i - kt_1 - \lfloor \frac{j}{t_1} \rfloor t_1, i + j - \lfloor \frac{j}{t_1} \rfloor t_1} \\ m'_{kt_1+j, (i+kt_1)} &= M_{i + kt_1 - kt_1 - \lfloor \frac{j}{t_1} \rfloor t_1, i + kt_1 + j - \lfloor \frac{j}{t_1} \rfloor t_1} \\ &= M_{i - \lfloor \frac{j}{t_1} \rfloor t_1, i + kt_1 + j - \lfloor \frac{j}{t_1} \rfloor t_1} \end{aligned}$$

leading to

$$P_i = \sum_{k=0}^{t_2-1} \sum_{j=0}^{t_1-1} m'_{kt_1+j, (i+kt_1)} x_{(i+kt_1)+j} = \sum_{k=0}^{t_2-1} \sum_{j=0}^{t_1-1} M_{i - \lfloor \frac{j}{t_1} \rfloor t_1, i + kt_1 + j - \lfloor \frac{j}{t_1} \rfloor t_1} x_{(i+kt_1)+j}$$

. Noticing that the downward rounded fraction  $\lfloor \frac{j}{t_1} \rfloor$  vanishes in a sum with  $j$  running from 0 to  $t_1 - 1$ , we can simplify to

$$P_i = \sum_{k=0}^{t_2-1} \sum_{j=0}^{t_1-1} M_{i, i + kt_1 + j} x_{i + kt_1 + j}$$

which contains two sums running to  $t_1$  and  $t_2$  respectively, containing an expression of the form  $k \cdot t_1 + j$ , which allows us to condense the nested sums into one single summation expression, as

$$\sum_{k=0}^{t_2-1} \sum_{j=0}^{t_1-1} f(kt_1 + j) = \sum_{l=0}^{t-1} f(l)$$

indeed catches every single value  $l \in \{0, 1, 2, \dots, t = t_1 \cdot t_2\}$  with  $l = kt_1 + j$ . In summary, we obtain

$$\begin{aligned} P_i &= \sum_{k=0}^{t_2-1} \sum_{j=0}^{t_1-1} M_{i, i+kt_1+j} x_{i+kt_1+j} \\ &= \sum_{l=0}^{t-1} M_{i, i+l} x_{i+l} = \sum_{\nu=0}^{t-1} M_{i, \nu} x_{\nu} \\ &= \{M\mathbf{x}\}_i \end{aligned}$$

which indeed equals the conventional definition of a matrix-vector product.  $\square$

Note that the optimized matrix-vector multiplication only requires  $t_1 + t_2$  as we can store the  $t_1$  inner rotations of the vector  $x$  for the upcoming evaluations. For larger matrices and vectors (larger  $t$ ),  $t_1 + t_2$  are indeed much smaller than the conventional number of required rotations in the Diagonal or Hybrid method for instance which was the point of this modification in the first place.

### 3.4 Polynomial Evaluation

From the implementation perspective, there are three properties to watch out for when working with SEAL ciphertexts:

1. Scale (retrieved using `x.scale()`)

Scale has nothing to do with noise. "Scale out of bounds" can appear even if noise is extremely low. Although repeated multiplication of a ciphertext by a plaintext will slowly increase the noise, it is not the reason why you see "scale out of bounds". "Scale out of bounds" error specifically means that the scale of a ciphertext or plaintext is larger than the product of all elements in `coeff_modulus`. If you perform multiplications without rescaling, you can quickly see this error. The more rescaling you perform, the less elements will be left in `coeff_modulus`. Even if you managed to have the same scale in a ciphertext after every multiplication and rescale, eventually the `coeff_modulus` can be too small to accommodate another multiplication.

Can be adjusted with: `evaluator.rescale_inplace()`

2. Encryption Parameters (retrieved using `x.parms_id()`)

Can be adjusted with: `evaluator.mod_switch_to_inplace()`

3. Ciphertext Size (retrieved using `x.size()`)

Can be adjusted with: `evaluator.relinearize_inplace()`

**Multiplication** Each time one multiplies two ciphertexts, the scales multiply (logarithmically, they add up, i.e. the bits are added together). The chain index reduces by 1. The chain index of an encoded ciphertext depends on the coeff moduli. There must be enough bits remaining to perform the multiplication, namely  $\log_2(\text{scale})$  bits.

**Addition** The scales must be the same, but luckily they will not change.

## 3.5 Transparent Ciphertext

The problem is that you are subtracting a ciphertext from itself. This kind of operation results in a ciphertext that is identically zero; this is called a transparent ciphertext. Such a transparent ciphertext is not considered to be valid because the ciphertext reveals its underlying plaintext to anyone who sees it, even if they don't have the secret key. By default SEAL throws an exception when such a situation is encountered to protect you from a problem you may not have noticed. If you truly know what you are doing and want to enable the creation of transparent ciphertexts, you can configure SEAL [...]. ([Laine 2020](#))

'transparent ciphertexts (a.k.a. ciphertexts whose second polynomial is zero) are malformed and do not need the secret key to decrypt'

# Chapter 4

## Results

4.1 Performance / Runtime Benchmarks

4.2 Communication Overhead

4.3 Accuracy



# Chapter 5

## Conclusion

# Notation

$\oplus$  exclusive-or (XOR)

9

# Acronyms

AES	Advanced Encryption Standard	9
FHE	Fully Homomorphic Encryption	6
HE	Homomorphic Encryption	6
ML	Machine Learning	8

# Bibliography

- Bishop, Christopher M. and Nasser M. Nasrabadi (2007). *Pattern Recognition and Machine Learning*. Vol. 16, p. 049901.
- Daemen, Joan and Vincent Rijmen (2002). *The Design of Rijndael: AES – The Advanced Encryption Standard*. Information Security and Cryptography. Springer. ISBN: 3-540-42580-2. DOI: [10.1007/978-3-662-04722-4](https://doi.org/10.1007/978-3-662-04722-4).
- Dobraunig, Christoph, Lorenzo Grassi, Lukas Helming, Christian Rechberger, Markus Schafneger and Roman Walch (2021). ‘Pasta: A Case for Hybrid Homomorphic Encryption’. In: *IACR Cryptol. ePrint Arch.* 2021, p. 731.
- Goldwasser, Shafi (2018). ‘From Idea to Impact, the Crypto Story: What’s Next?’ In: URL: <https://www.youtube.com/watch?v=culuNbMPP0k> (visited on 01/03/2022).
- Halevi, Shai and Victor Shoup (2018). *Faster Homomorphic Linear Transformations in HElib*. Cryptology ePrint Archive, Report 2018/244. <https://ia.cr/2018/244>.
- Hornik, Kurt, Maxwell B. Stinchcombe and Halbert L. White (1989). ‘Multilayer feedforward networks are universal approximators’. In: *Neural Networks* 2, pp. 359–366.
- Juvekar, Chiraag, Vinod Vaikuntanathan and Anantha P. Chandrakasan (2018). ‘Gazelle: A Low Latency Framework for Secure Neural Network Inference’. In: *CoRR* abs/1801.05507. arXiv: [1801.05507](https://arxiv.org/abs/1801.05507). URL: <http://arxiv.org/abs/1801.05507>.
- Laine, Kim (2020). *Result Ciphertext is Transparent*. GitHub. URL: <https://github.com/microsoft/SEAL/issues/224#issuecomment-702516479> (visited on 04/03/2022).
- LeCun, Yann and Corinna Cortes (1998). *The MNIST database of handwritten digits*. URL: <http://yann.lecun.com/exdb/mnist/>.
- Nüst, Daniel, Vanessa Sochat, Ben Marwick, Stephen J. Eglén, Tim Head, Tony Hirst and Benjamin D. Evans (2020). ‘Ten simple rules for writing Dockerfiles for reproducible data science’. In: *PLOS Computational Biology* 16.11, e1008316. DOI: [10.1371/journal.pcbi.1008316](https://doi.org/10.1371/journal.pcbi.1008316).
- Shannon, Claude E. (1949). ‘Communication Theory of Secrecy Systems’. In: *Bell System Technical Journal* 28.4, pp. 656–715. DOI: [10.1002/j.1538-7305.1949.tb00928.x](https://doi.org/10.1002/j.1538-7305.1949.tb00928.x).

# List of Figures

- 2.1 Comparison of the Relu activation function vs. its Taylor expansion . . . . . 9
- 3.1 Image source: [Juvekar, Vaikuntanathan and Chandrakasan 2018](#) . . . . . 12

# List of Tables