Privacy-Preserving Moving Object Detection

Computer Science Tripos - Part II

Candidate Number

Department of Computer Science and Technology
University of Cambridge

Friday 13 May, 2022

Declaration of Originality

. . . DECLARATION OF ORIGINALITY . . .

Proforma

. . . PROFORMA . . .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the modern world, computers have improved almost every aspect of our lives. Recently, home security has become the latest target of the technology revolution. Companies like Ring [32] and Eufy [10] offer IoT devices like doorbells and cameras to allow their customers to monitor their property 24/7. On top of traditional surveillance, these companies also provide software solutions to monitor the footage recorded by their devices and interpret it. For example, a doorbell may recognise who is at the front door and allow them to enter, or alert the user to the presence of a stranger if it doesn't. However, the computational intensity of these inferences means footage must be transferred from the devices to more powerful servers.

In order to preserve privacy, video is encrypted before it is sent to the server. However, the footage must be decrypted when the inference algorithms are executing. This is an immediate privacy concern. Having the ability to decrypt the footage exposes the opportunity for employees of these companies to access constant surveillance of peoples' homes. The possibilities for exploitation are endless. Malicious actors could use this information to monitor people's location, appraise their belongings, or use the contents of footage for extortion, to name a few. Homomorphic Encryption may provide a solution to this.

Homomorphic Encryption (henceforth HE) is a cryptographic method of encrypting data such that mathematical operations can be performed on encrypted data, or *ciphertext*, itself, rather than on the raw data, or *plaintext*. For example, consider the operation 3×5. In a traditional encryption scheme, the plain values 3 and 5 would be multiplied before encrypting the result. Using a homomorphic scheme, the 3 and 5 can be encrypted, and the ciphertexts multiplied so that when the ciphertext is decrypted, the plaintext is 15. An open question is, can this technique be scaled to more complex algorithms, like those required for surveillance?

More specifically, is it possible to extract the moving objects from a frame of HE video data? Moving object detection, also known as *foreground extraction* or *background subtraction*, is fundamental to modern surveillance systems. Detecting when, for example, somebody enters a property, allows the security systems to alert their owners, possibly pre-empting a break-in. To perform this analysis, the contents of a video must be modelled using a, usually probabilistic, function that allows significant changes in a pixels' value to be discerned. The difficulty of this arises when accounting for environmental changes that cause numerical variation, such as light levels when moving from day to night or different weather conditions causing objects to distort.

1.2 Related Work

The lack of privacy caused by constant surveillance is not a new concern. There have been many attempts at solving video inference in the encrypted domain, but none are without flaws. For example, in 2013, Chu et al. [8] proposed an encryption scheme that supports real-time moving object detection, but this was quickly shown to suffer from information leakage, leaving it vulnerable to chosen-plaintext attacks[1]. Similarly, in 2017, Lin et al. [26] proposed a different encryption scheme to achieve the same goal by only encrypting some of the bits in each pixel, but this is unprotected against steganographic[2] attacks. Therefore, while research has been able to solve the weaknesses in privacy, it is yet to offer a solution that also preserves security against adversaries directly attacking the encryption, making them useless to real-world applications.

Likewise, researchers have been investigating inference using HE for many years. In 2012, Graepel et al. [15] introduced machine learning in the HE domain. Dowlin et al. [15] built upon this when they developed the CryptoNets model for deep learning with HE in 2016. However, deep learning neural networks are considered overly complex for moving-object detection. Instead, GMMs are the most widely used technique for background modelling. There is much less research into this area of unsupervised learning within the HE domain. The best example appears to be when, in 2013, Pathak and Raj [31] proposed a HE implementation of a GMM for audio inference. But there does not seem to be any investigations linking HE and GMMs to video analysis.

It appears that the most prevailing explanation for this lack of research is HE's inapplicability to real-time applications, due to its high computational complexity. While this may be true now, it is important to acknowledge that advances in computing capability will reduce the relative difficulty of HE operations. Consequently, more insight into its applicability will become increasingly valuable, as suggested by the trend in the growing popularity of HE research. This dissertation attempts to offer some beginnings to this insight as it attempts to find the constraining limitations of current HE implementations with respect to surveillance.

1.3 Aims and Contributions

This dissertation documents the design and implementation of a potential solution to the questions posed in §1.1, while attempting to follow the constraints restricting the aforementioned real-world systems. In particular, the contribution of the work is:

- The creation of a client-server application simulating the device-server stack utilised by existing products, allowing secure transmission of video data from client to server and back again after performing inference.

[1]A *chosen-plaintext attack* is a scenario in which an adversary can encrypt plaintexts of their choosing, and analyse the corresponding ciphertext in an attempt to break the encryption.

[2]*Steganography* describes the technique of information hiding. Like cryptography, steganography attempts to prevent adversaries from reading messages. Unlike cryptography, where the existence of a message is known but its contents are not, steganography attempts to hide the message's existence.

- The use of Microsoft's Secure Encrypted Arithmetic Library (SEAL) [24] to integrate the CKKS HE scheme [7] for encrypting videos while they are away from the client.
- The implementation of a series of algorithms for enabling private and plain inference of video data to extract moving objects by producing a mask that can be applied to videos in the clear by the client.
- An investigation of Gaussian Mixture Models (GMMs) for HE encrypted background subtraction, beginning with the work by Stauffer and Grimson [35] then moving into more general Expectation-Maximisation GMM algorithms [SOURCE?].
- As an extension, the fabrication of a CKKS implementation from scratch, called MeKKS, based on the Homomorphic Encryption for Arithmetic of Approximate Numbers paper by Cheon et al. [7, 6] to improve understanding of HE.
- A demonstration of the efficacy of the above solutions using timing, accuracy, and *(hopefully)* energy usage data to compare inference of CKKS and MeKKS solutions to plain videos, highlighting the advantages of the MeKKS implementation being targeted to this application over the more generic CKKS.

Chapter 2

Preparation

This chapter discusses the preparatory work done before beginning the project's implementation. §2.1 introduces useful preliminary information and background, §2.2 discusses the methodologies used when approaching the design of the project's implementation, and §2.3 provides an overview of the foundations of the project.

2.1 Preliminaries

This section is dedicated to introducing the fundamental underlying concepts, terminologies, and notations fundamental to this project. The project spans two domains of computer science: cryptography and unsupervised machine learning. These are vast, active research areas. Therefore, for brevity, only concepts essential to understanding this dissertation will be presented - links to further resources may be provided for the sake of brevity.

2.1.1 Threat Model

The design of this project considers the Machine Learning as a Service (MLaaS) framework, in which users first send their data to a server where machine learning inference is performed, and the results are returned. In particular, the transfer of surveillance video data to be analysed by security companies. Suppose that a subscriber to one of these services, Alice, sets up a camera to record activity at her front door. In this scenario, there are two critical threats: (*i*) an adversary, Eve, may eavesdrop on the data while it is being transmitted between the client and server, and (*ii*) an employee, Mallory, of the MLaaS provider may perform unauthorised accesses on the data while it is being stored by the server. Figure 2.1 illustrates this succinctly. The first threat can be mitigated easily enough using cryptographic protocols such as TLS [36]. However, the second threat is much more difficult to defend against, particularly because data must usually be decrypted for inference to be performed [2].

Fortunately, the use of HE can mitigate both of these risks. Firstly, HE is a secure cryptographic encryption scheme, so using it to encrypt data during transmission is sufficient to thwart eavesdropping adversaries. Secondly, HE allows computation to be performed on the data without decryption, so it can prevent the exploitation of plain data.

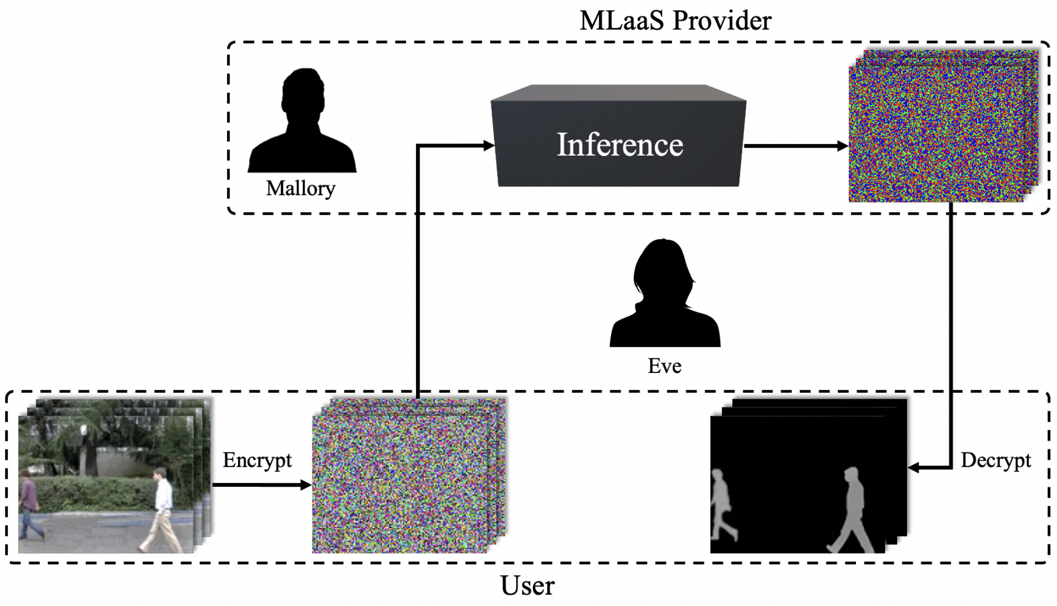


Figure 2.1: A graphical representation of the threat model. Eve is an eavesdropper able to listen to communications between user an service provider. Mallory is a malicious individual within the service provider able to access users' video data. HE is able to obstruct both Eve and Mallory, preventing them from discovering the contents of the user's video.

2.1.2 Homomorphic Encryption

Introduction

There are two broad categories of cryptographic encryption schemes: private-key (symmetric) and public-key (asymmetric). While both can be applied to HE, this dissertation will address public-key encryption because it is the technique adopted by all HE schemes used in the project.

A public-key encryption scheme is defined by a triple of functions $\Pi = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$. `KeyGen` is a function used to generate a *public key* (PK) and *private key*[1] (SK) such that $(\texttt{PK}, \texttt{SK}) \leftarrow \texttt{KeyGen}(1^l)$, where the security parameter, l, measures how hard it is for an adversary to break the scheme[2]. Denoting the space of all possible plaintext messages as $\mathcal{M}$ and ciphertext messages as $\mathcal{C}$, a message $m \in \mathcal{M}$ is encrypted into its corresponding ciphertext $c \in \mathcal{C}$ by $c \leftarrow \texttt{Enc}_{\texttt{PK}}(m)$. Similarly c is decrypted back into m by $m \leftarrow \texttt{Dec}_{\texttt{SK}}(c)$.

In order to extend Π into a HE scheme, a fourth function $\texttt{Eval}(f, c_1, \ldots, c_n)$ must be introduced. The evaluation function, `Eval` applies a Boolean circuit, f, to the ciphertext arguments, $c1, \ldots, c_n$ such that, for all arguments, it holds that

$$\texttt{Dec}_{\texttt{SK}}(\texttt{Eval}(f, c_1, \ldots, c_n)) = f(m_1, \ldots, m_n) \tag{2.1}$$

where $m_1, \ldots, m_n$ are the plaintext equivalents of $c_1, \ldots, c_n$. This is, perhaps, better illustrated by Figure 2.2 below.

[1]The *private key* is referred to as a *secret key* by some literature. While these terms are equivalent, general convention is to use *secret key* in relation to symmetric encryption, and *private key* when discussing asymmetric - a practice that this dissertation will follow.

[2]An l-bit security parameter would require an expected 2^l attempts to guess the keys.

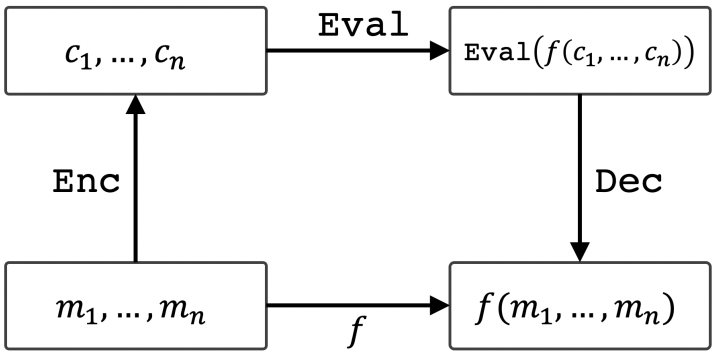


Figure 2.2: Homomorphic Encryption

Theoretically, a *fully* homomorphic scheme[3] allows the evaluation of Boolean circuits indefinitely. However, in practice, the time complexity of operations means many schemes are *levelled* homomorphic schemes. This means they only support operations up to a *bounded depth* - that is, only a predefined number of circuits can be applied to a ciphertext before the plaintext becomes irrecoverable. The maximum depth is a critical factor when applying HE to practical problems because it significantly limits the scope of supported algorithms.

Ring Learning with Errors

For an encryption scheme to be *perfectly secure*, a ciphertext must provide no additional information about its plaintext (see Equation 2.2). In other words, the probability of generating a given ciphertext from a particular plaintext is independent of the plaintext (see Equation 2.3). The two equations below can be shown to be equivalent using Bayes' rule.

$$\mathbb{P}(M = m \mid C = c) = \mathbb{P}(C = c) \tag{2.2}$$

$$\mathbb{P}(C = c \mid M = m) = \mathbb{P}(M = m) \tag{2.3}$$

for all $m \in \mathcal{M}$, $c \in \mathcal{C}$.

While it is possible to create perfectly secure encryption schemes, they are impractical in real applications[4]. Therefore, *computational security* is considered sufficient. Relying on the hardness of certain mathematical problems, computational security means that an encryption scheme is *practically unbreakable*. That is, the most efficient known algorithm for breaking a cipher would require far more computational steps than an attacker would be able to perform, regardless of the hardware available to them. An example of this is the RSA encryption scheme which relies on the fact that there exists no known, efficient algorithm for computing the prime factors of a large number on a classical computer - the integer factorisation problem. In complexity theory, this problem falls into the set of $\mathcal{NP}$.

Similarly, the HE schemes used in this project rely on computational security

[3]The prefix *fully* derives from the existence of *partially* homomorphic schemes. A partially homomorphic scheme will only allow certain operations on the ciphertext - usually multiplication and division - and have existed for many years. Some examples of partially homomorphic schemes include RSA, ElGamal, and Paillier encryption [30].

[4]For example, the One-Time Pad [37]

rather than perfect security. More specifically, they utilise the hardness of the *ring learning with errors* (henceforth RLWE) problem introduced by Lyubashevsky et al. [27]. A polynomial-time reduction from the *shortest vector problem* to RLWE can be derived. Therefore, since the shortest vector problem is $\mathcal{NP}$-hard under the correct choice of parameters, it is safe to rely on RLWE for computational security.

RLWE considers the mathematical objects, *rings*. To understand *rings*, *groups* must first be understood. A group $(\mathbb{G}, \bullet)$ is a set, $\mathbb{G}$, and an operator, $\bullet : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$, such that the following properties hold:

- **Closure**: $a \bullet b \in \mathbb{G}$ for all $a, b \in \mathbb{G}$.
- **Associativity**: $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all $a, b, c \in \mathbb{G}$.
- **Neutral Element**: there exists an $e \in \mathbb{G}$ such that for all $a \in \mathbb{G}$, $a \bullet e = e \bullet a = a$
- **Inverse Element**: for each $a \in \mathbb{G}$ there exists some $b \in \mathbb{G}$ such that $a \bullet b = b \bullet a = e$

If $a \bullet b = b \bullet a$ for all $a, b \in \mathbb{G}$, the group is called **commutative** (or **abelian**). If there is no inverse element for each element, $(\mathbb{G}, \bullet)$ is a **monoid** instead.

From this, a *ring* is defined as $(\mathbf{R}, \boxplus, \boxtimes)$, where $\mathbf{R}$ is a set, $\boxplus : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$, and $\boxtimes : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$, such that

- $(\mathbf{R}, \boxplus)$ is an abelian group.
- $(\mathbf{R}, \boxtimes)$ is a monoid.
- $\boxplus$ and $\boxtimes$ are distributive - for all $a, b, c \in \mathbf{R}$, $a \boxtimes (b \boxplus c) = (a \boxtimes b) \boxplus (a \boxtimes c)$ and $(a \boxplus b) \boxtimes c = (a \boxtimes c) \boxplus (b \boxtimes c)$.

If $a \boxtimes b = b \boxtimes a$ then it is a **commutative** ring, but this is not necessary for rings generally. One example of a ring is $(\mathbb{Z}[x], +, \times)$.

Specifically, the RLWE problem is concerned with the ring formed by the set of polynomials modulo $\Phi(X)$ that also have coefficients in $\mathbb{Z}_q$[5]. Known as a *quotient ring*, this can be denoted by $\mathcal{R}_q = \mathbb{Z}[X]/(\Phi(X))$, where $\Phi(X)$ is an *irreducible polynomial* - a polynomial which cannot be factored into two non-constant polynomials.

Informally, RLWE describes the problem of finding an unknown $s \in \mathcal{R}_q$ given a vector of polynomials computed using s and some sampled errors. Consequently, an encryption scheme can be created such that, after encoding a plaintext vector, $\mathbf{v}$, as a list of polynomials and then using a secret polynomial to convert this list to a polynomial, it is infeasible to recover $\mathbf{v}$ in polynomial time.

The HE schemes discussed in this dissertation rely on the RLWE problem to assert *indistinguishable encryptions under a chosen-plaintext attack* (IND-CPA) security. Fundamentally, any encryption scheme is IND-CPA secure if, when attacked by a probabilistic, polynomial-time adversary, the chances of correctly deciding which of two plaintexts a particular ciphertext corresponds to is even. Therefore, in practice, even if an adversary has access to an encryption *oracle* that will encrypt any data an adversary provides, the adversary's chances of calculating the correct plaintext when given a ciphertext should be no better than if they were randomly guessing.

[5]$\mathbb{Z}_q$ is the set of integers modulo q. For example, $\mathbb{Z}_7 = \{0, 1, 2, 3, 4, 5, 6\}$.

Practically, HE schemes choose a *cyclotonic polynomial* for $\Phi(X)$, where the n-th cyclotonic polynomial, $\Phi_n(X)$, is defined as

$$\Phi_n(X) = \prod_{k \in [1,n];\ gcd(k,n=1)} X - e^{\frac{2i\pi k}{n}} \tag{2.4}$$

In order to speed up computation, n is selected to be an even power of two. Consequently, $\Phi_n(X) = X^{\frac{n}{2}} + 1$. This allows the *number theoretic transform* (NTT)[6]. The advantage of this is that it can be easily accelerated using hardware [22].

The sampled errors used when deriving ciphertexts implies almost exponential growth in error in the number of multiplications applied. The limited depth property of levelled HE schemes is a direct result of this. However, the relative growth size can be reduced by increasing the modulus q. Although this is not without risks. If a polynomial degree of size $n/2$ is used, efficient attacks exist against the RLWE problem for a small value of q [1]. Therefore, a fundamental trade-off is introduced between the supported depth of multiplication and the security level.

2.1.3 Moving Object Detection

Introduction

A discussion of *image segmentation* has been well established in the fields of digital image processing and computer vision research. The problem describes the process of partitioning a digital image into multiple regions, represented as sets of pixels. The goal of segmentation is to simplify the representation of an image so that it is easier to analyse. For example, to locate objects, or boundaries, in an image. More precisely, image segmentation is the process of labelling each pixel of an image such that all pixels sharing a particular characteristic are assigned the same label [34].

There are two broad categories of segmentation techniques: *semantic segmentation* and *instance segmentation*. Semantic segmentation is an approach for grouping objects based on predefined categories. For example, all people in an image may be labelled as “people”, all vehicles labelled “vehicles”, and all animals labelled as “animals” [16]. In contrast, instance segmentation provides a more refined categorisation of objects. This approach splits each category into separate occurrences. For example, each person in an image may be highlighted distinctly [40].

One application of image segmentation is *foreground extraction*, also known as *background subtraction*, which describes the techniques of segmenting an image into two groups: the foreground and the background, so that further processing can be applied. In order to perform background subtraction, the background of an image must be modelled so that changes in the scene can be detected. However, this can be difficult to achieve. Image data can be very diverse, with factors such as variable lighting, repetitive movements (like leaves, waves, and shadows) making robust models hard to develop.

Once a background subtraction model has been developed, it can be used to detect moving objects in videos. This is accomplished by comparing the foreground of the

[6] A specialisation of the discrete Fourier transform, the NTT is a generalisation of the Fast Fourier transform in the case of finite fields. [3]

current frame to the foreground of a reference frame and extracting the observed differences. There are several methods for achieving this. Below are the five algorithms investigated for this dissertation.

Frame Differencing

The most straightforward moving object detection algorithm, *frame differencing* works by iterating through the frames of a video in a single pass. First, a reference frame must be established as the background. There are several options for this. One approach is to store the first frame of the video. Another, more common option is to compare each frame to the frame directly before it. The advantage of this is that it will evolve, so if a new object is permanently added to the scene, it won't be included in the foreground forever[7]. However, there are disadvantages to this approach if an object is moving slowly enough to overlap with itself across frames. A balance can be found by periodically updating the reference frame or comparing each frame to one several before it, for example.

Each frame can be considered once the reference frame, B, has been selected. Denoting the frame at time t as f_t, the value of each pixel in B, $P(B)$, can be subtracted from the corresponding pixel in f_t, $P(f_t)$. This can be represented mathematically by Equation 2.5.

$$P(F_t) = P(f_t) - P(B) \tag{2.5}$$

where F represents the frames in the resultant video highlighting moving objects.

Mean Filter

A *mean filter* approach to moving object detection attempts to overcome the weaknesses of selecting a reference frame when performing frame differencing. Instead of taking a frame directly from the video, the value of B at time t is calculated using Equation 2.6.

$$B = \frac{1}{N}\sum_{i=1}^{N} f_{t-i} \tag{2.6}$$

where N is the number of preceding images included in the average, and f_t is the frame in the video at time t. N would depend on the video speed and the amount of motion expected in the video.

After B has been calculated at time t, the value of the resultant video F_t can be calculated using the same method as frame differencing, given by Equation 2.5.

Median Filter

Performing moving object detection using a median filter is almost identical to the mean filter method. The difference arises in how the reference frame is calculated. In this approach, the median of the preceding N frames is calculated instead of the mean.

Then, like the preceding methods, the moving objects are extracted by subtracting the reference frame from each frame in the video, according to Equation 2.5.

[7] For example, if a fence was added around someone's property or a car was parked in the scene, comparing to a static frame would mean these objects would always be highlighted by frame differencing, despite not moving

Gaussian Average

Wren et al. originally proposed fitting a Gaussian probabilistic density function to the most recent N frames [39]. Rather than storing a simple reference image that is subtracted from each frame in the video, this method stores a mean and variance value for each pixel in a video frame. The likelihood of a value of a particular pixel occurring can be calculated using this. It is assumed that the most likely value for a pixel will be equivalent to the background of a scene, so if an observed value is sufficiently unlikely according to its Gaussian distribution, it must be because a change has occurred in that portion of the frame. Therefore, that pixel is added to the foreground segment of the video.

A naïve approach to the Gaussian Average method would be to iterate through each frame in the video, calculating the mean and variance for each pixel and evaluating the likelihood from scratch every time. This would have quadratic complexity since, for each frame, N calculations per pixel would have to be performed. A more efficient algorithm would utilise a cumulative function for the mean and standard deviation that can be updated in constant time. Consequently, this would have linear complexity because only a single calculation would need to be performed for each pixel in each frame. This approach would update the mean using Equation 2.7, and the variance using Equation 2.8.

$$\mu_t = \begin{cases} f_0 & \text{if } t = 0 \\ \alpha f_t + (1-\alpha)\mu_{t-1} & \text{otherwise} \end{cases} \tag{2.7}$$

$$\sigma_t^2 = \begin{cases} c & \text{if } t = 0 \\ d^2\alpha + (1-\alpha)\sigma_{t-1}^2 & \text{otherwise} \end{cases} \tag{2.8}$$

where α determines the size of the *temporal window* used to fit the Gaussian model[8], $d = |f_t - \mu_t|$ gives the Euclidean distance from the pixel to the mean, and c is some constant defined by the model creator.

From these models, the foreground can be extracted according to Equation 2.9,

$$\frac{|f_t - \mu_t|}{\sigma_t} > k \tag{2.9}$$

where k is a constant that the model creator can tune to achieve optimal results.

A variant of this method exists where the mean and variance are only updated if a pixel is believed to be in the background. This prevents the model from becoming skewed if there is lots of movement in the frame. However, it has severe limitations. For example, it only works if the image is initially entirely background, and it cannot cope with gradually changing backgrounds.

Gaussian Mixture Models

Stauffer and Grimson proposed *Gaussian mixture models* (henceforth GMMs) for moving object detection in 1999 [35]. GMMs are probabilistic models that represent

[8] α acts similarly to a decay factor, weighting the most recent frames as more impactful on the model. Eventually, an old frame will be weighted so insignificantly that its impact is negligible. Therefore, it determines how far into the past the model uses to predict future pixels, hence the 'temporal window'.

the presence of normally distributed subpopulations within an overall population. They are particularly useful because they don't require the subpopulation of a data point to be identified. Instead, subpopulations are learned automatically, constituting a form of unsupervised machine learning.

A simple application of GMMs is in modelling human heights. Typically, two Gaussian distributions will be used: one for males and one for females. Consequently, given just height data, with no gender assignments, it can be assumed that the distribution of all heights should follow the sum of two scaled and shifted Gaussian distributions. A model that can make this assumption on its own (or unsupervised) is an example of a GMM. In general, GMMs can be applied to more than two components.

There are two types of parameters necessary for GMMs, the *component weights*, and the component *means* and *variances*. For a GMM with N components, the i^{th} component has μ_i and variance σ_i in the *univariate case*, and mean μ_i and a covariance matrix Σ_i in the *multivariate case*. The component weights, ϕ_k for component k, are constrained by the equation $\sum_{i=1}^{K} \phi_i = 1$. If the component weights aren't learned, they are known as an *a-priori*[9] distribution over components such that $\mathbb{P}(x \text{ generated by component k}) = \phi_k$. If the component weights are learned, they are known as *a-posteriori*[10] estimates of the component probabilities given the data.

To estimate a GMM's parameters, for a known number of components N, *expectation maximisation* is used. This means assigning each data point to a category such that the total probability of the data given the model parameters is maximised. The iterative algorithm consists of two steps. Firstly, the expectation step (or E-step) calculates the expectation of assigning each data point to each component, given the GMM's parameters. Secondly, the maximisation step (or M-step) updates the values of each parameter to maximise the expectations. These two steps repeat until the algorithm converges. Informally, this works because knowing the component assignment for each data point makes solving for the parameters easy, and knowing the parameters makes inferring the probability of a component given the data point easy[11]. Therefore, by alternating which values are assumed known, maximum-likelihood estimates of the unknown values can be efficiently calculated.

Once the EM algorithm has completed, the fitted model can be used for inference. There are two common uses of inference: *density estimation* and *clustering*. This dissertation will focus on clustering because it is most useful for moving object detection. The posterior component assignment probabilities can be estimated using Bayes' theorem combined with the GMM parameters. Knowing the component that a data point most likely belongs to provides a way to group the points into clusters. In the scenario of moving object detection, this would be two clusters: the foreground and the background.

2.2 Project Strategy

[9] A probability derived purely through deductive reasoning.

[10] From Bayesian statistics, referring to conditional probability $\mathbb{P}(A|B)$.

[11] The E-step corresponds to the latter case, and the M-step corresponds to the former.

2.2.1 Requirements Analysis

The requirements for this project are listed below. The nature of the project required the development of theoretical knowledge before implementation began, and, as such, the requirements subtly evolved as understanding matured. The original requirements are given in Appendix A for comparison. The final requirements have been grouped into two categories. The first, labelled *A*, are the core requirements deemed essential to the project's success. The second, labelled *B*, are considered extensions, aiming to improve understanding of the components used in the core implementation or further the investigation into the applications of HE in surveillance.

Core

`A1`: *Implement a client-server application allowing videos to be homomorphically encrypted and transmitted in both directions.*

This component is vital because it provides the foundation for implementing and integrating all other components. It is also essential to emulate the MLaaS software stack. While conceptually simple, the nature of HE data adds many challenges to transferring videos, forming a significant portion of the investigation into HE applicability.

`A2`: *Implement background subtraction models that can extract moving objects from homomorphically encrypted videos.*

This requires designing and implementing the five moving object detection algorithms detailed in §2.1.3 so that they can act on HE data.

`A3`: *Evaluate the accuracy of HE inference to investigate its applicability to real systems.*

This involves analysing different metrics to understand the efficacy of moving object detection on HE data. Comparisons can be made between inference methods and between plain and encrypted data.

Extensions

`B1`: *Implement a bespoke HE scheme and integrate it into the core application, providing the same functionality as the established scheme already used.*

While this implementation will likely have worse performance than existing implementations, it will offer helpful insight into the inner workings of HE. Also, it may provide opportunities for optimisations for this specific application.

`B2`: *Analyse the security of the encryption schemes used in the project.*

This will be useful in ensuring HE can overcome both security and privacy concerns of existing surveillance solutions and doesn't accidentally introduce insecurities that could allow adversaries to extract information.

`B3`: *Implement an object recognition algorithm acting on HE data using neural networks.*

Implementations like Cryptonets [9] have demonstrated the application of neural networks to HE. This would allow further services offered by surveillance companies to be emulated and reveal a greater insight into the limitations of HE in video analysis.

2.2.2 Methodology

A Waterfall development methodology was adopted for this project [33]. The requirements were detailed and unambiguous, so the project lent itself to a structured methodology, not requiring the flexibility of an iterative approach. The stages of the model are detailed below.

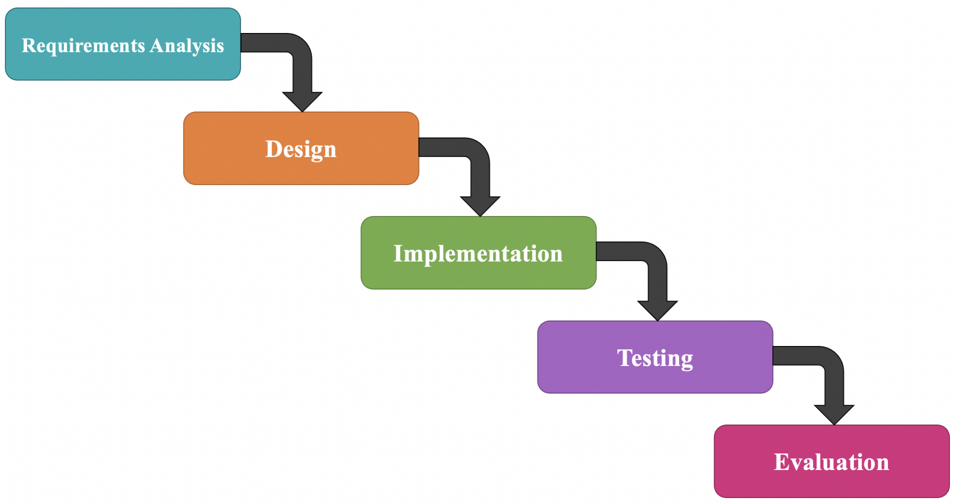


Figure 2.3: The Waterfall Development Model

The results of the *requirements analysis* stage have been detailed in §2.2.1. This stage is where most of the research was performed so that the project's design would be better informed.

The *design* phase incorporates expanding on the requirements into a physical project; this includes, for example, creating the class diagram shown in Figure **??**.

The *implementation* and *testing* stages were intertwined where possible in order to promote a test-driven approach to development. This was made easier by the object-oriented methodology (see §**??**) and unit testing practices (see §**??**) adopted during the design stage.

The *evaluation* stage replaces the *maintenance* stage of the traditional Waterfall model because it is unsuitable for this project. This stage involves running experiments to evaluate the project.

When working on the extensions, an iterative model was more appropriate. For example, extension `B1` could be easily split into distinct modules, so they could be developed then tested in a repeating process. Therefore, development transitioned to an Agile model, depicted in Figure 2.4.

A Gantt chart of the project's timeline is shown in Figure 2.5.

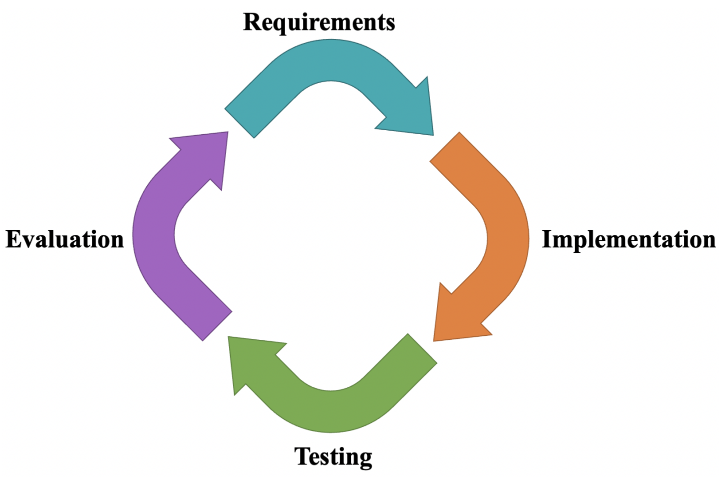


Figure 2.4: The Agile Development Model

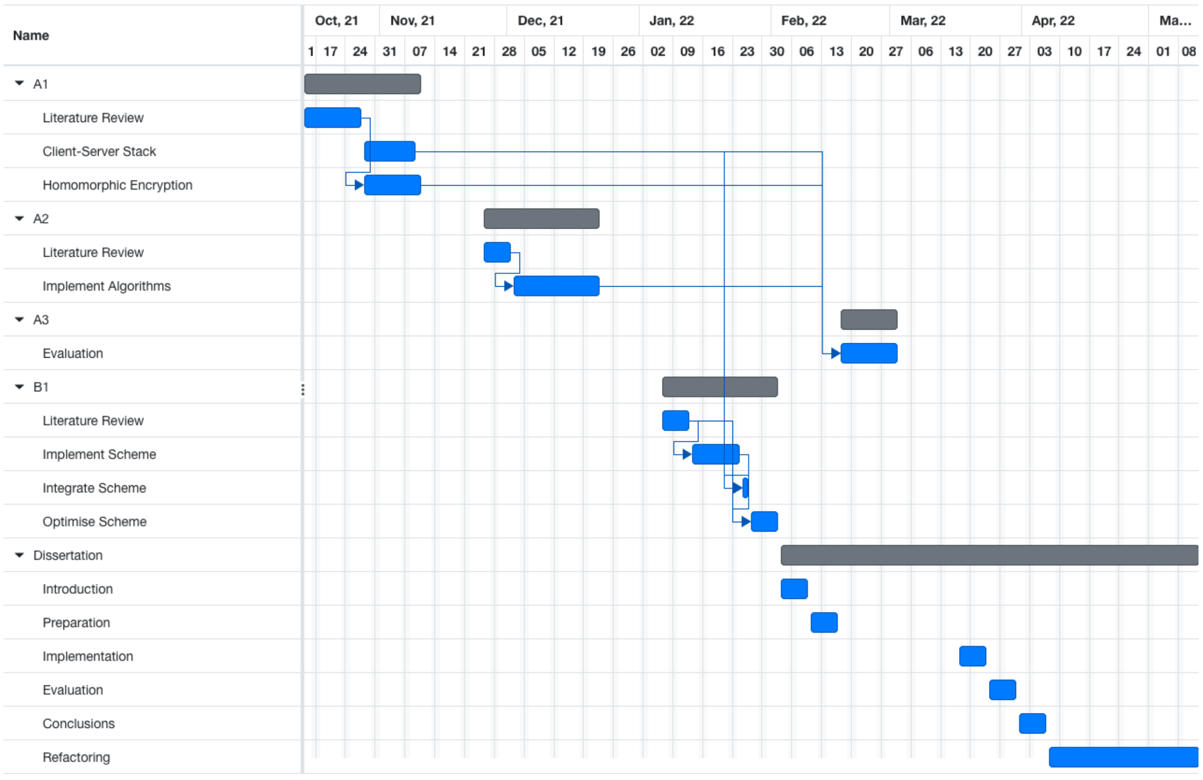


Figure 2.5: Project Timeline

2.2.3 Testing

Challenges

Unlike traditional software engineering, machine learning does not provide precise criteria against which the correctness of an implementation can be verified. The models used for background subtraction are probabilistic, so the outputs cannot be precisely predicted. Consequently, a variety of testing methodologies were required.

Unit Tests

Designed for testing atomic units of source code, unit testing takes advantage of the independent nature of components written following object-oriented principles to test functions in isolation. It does this by providing known expected, boundary, and erroneous data and ensuring the results of a function match the expected. Unit tests can be automated, making them easy to run repeatedly as changes to the source code are made, ensuring errors aren't introduced.

Unit tests were particularly useful when developing a HE scheme from scratch. They verified the correctness of the encoding, encryption, and decryption functions and the HE Boolean circuits automatically and repeatedly.

Integration Tests

While similar to unit tests, integration tests increase the scope of functionality covered by each test. The goal of integration testing is to ensure separate modules interact correctly. Once unit testing has been completed, these tests take the verified modules, group them into larger aggregates, and provide expected, boundary, erroneous data to ensure the output is correct.

Integration testing was useful in verifying that the software stack functioned correctly. For example, ensuring the client and server communicated correctly. While some integration testing can be automated, this project relied on manual testing because of requirements such as the client and server needing to be run independently.

Manual Verification

Manual verification was used to overcome the challenges of testing the background subtraction models. Since the project involves video data, human inspection provides a good intuition of whether or not a background has been correctly removed. If a more detailed analysis is required, pixel values can be compared to check for expected results, or verify consistency across multiple tests[12].

2.3 Starting Point

2.3.1 Knowledge and Experience

Prior to beginning the project, the following Tripos courses that considered similar themes had been completed: *Scientific Computing*, *Machine Learning and Real-world Data*, *Software and Security Engineering*, *Concurrent and Distributed Systems*, *Data Science*, *Computer Networking*, and *Security*. The Part II course, *Cryptography* was also useful in understanding the theoretical underpinnings of encryption.

However, it should be noted that HE is not included in the scope of the Cryptography course, so the theory was learned independently of Tripos studies. The study of applied HE is sparsely documented, particularly with modern schemes. Therefore, most understanding came from academic papers; notably [7] and [24]. Although, articles such as [5] were more useful for foundational knowledge.

[12]While humans are able to perform this verification, simple Python scripts are usually written to make testing more efficient.

Similarly, there was little mention of computer vision artificial intelligence in Tripos, so most understanding came from independent research. For example, academic papers such as [35] and [23] were helpful, particularly when considering privacy-preserving computer vision. Some understanding also came during a summer internship completed in the field of object recognition deep learning.

2.3.2 Tools Used

Programming Languages

All of the code written for this dissertation was written in *Python* [13]. The main reasons for this were the large machine learning ecosystem, ease of use, and ease of debugging. Consequently, it was best suited to the project's tight schedule since it should allow for quick implementation.

However, it must be acknowledged that Python is not a language traditionally used for cryptographic applications. Usually, lower-level, faster languages like C++ are favoured. Since the original focus of the project was investigating the efficacy of moving object detection in the HE domain, the speed of execution was not prioritised over the speed of implementation.

Software Development

Visual Studio Code [29] development environment was used for writing code because of support for Python as well as a wide variety of plugins that allow integration of other valuable tools such as ESLint. In addition, *Git* [14] and *GitHub* [19] were used for version control and source code management, as well as storing a backup of source code. *OneDrive* [28] was also used to hold another backup, for safety.

Encryption Schemes

The project focuses on HE schemes based on the RLWE problem. The main reason for this was the abundance of academic literature discussing them. In particular, the CKK scheme [7] was selected because it supports representing real numbers[13]. However, the project is designed in such a way that any HE encryption scheme can be substituted in CKKS's place, as long as it follows the same API.

Libraries

The project uses Microsoft's SEAL library [24], which provides a C++ implementation of the CKKS scheme. This was chosen because of the extensive optimisations that have been applied. In particular, SEAL uses a residue-number-system variant of CKKS to support large plaintext moduli. The SEAL API was integrated using a Python wrapper library [17].

Datasets

There were two publicly available datasets used in this project:

[13]As opposed to, for example, the BFV scheme, which only supports integers [4, 11].

- The Moving-MNIST dataset contains ten thousand sequences each of length twenty frames showing two handwritten digits from the standard MNIST dataset moving in a 64×64 pixel frame. This is a relatively simple dataset to perform moving object detection on because it only contains white objects on a black background. Therefore, it was useful in providing a baseline for the performance of the inference algorithms. [38, 25].
- The LASIESTA dataset contains a variety of sequences showing objects moving across static backgrounds in a range of conditions. Specifically designed to evaluate segmentation algorithms, this dataset provides a more realistic example of surveillance video. Therefore, this dataset can provide a truer evaluation of moving object detection in the HE domain. [18].

Licensing

All software dependencies in this project use permissive libraries that allow their code to be used without restrictions. The same is true for the datasets. Table 2.1 gives the specific licenses.

Dependency	**Licence**

`Multiprocessing`
`NumPy`
`SymPy`
	3-Clause BSD [20]

`Microsoft SEAL`
`SEAL-Python`
	MIT [21]

`Matplotlib`
`Python`
`Tkinter`
	PSFL [12]

Table 2.1: Licenses

2.3.3 Computer Resources

The original project proposal mentioned that external computational resources might have been required during the implementation phase, such as AWS or Microsoft Azure. However, the project was entirely developed, tested, and evaluated on a MacBook Pro laptop. The specifications are listed below, in Table 2.2.

Processor	
CPU	8 Cores
GPU	14 Cores
Neural Engine	16 Cores
Memory Bandwidth	200 GB/s
Memory	
RAM	32 GB (unified memory)

Table 2.2: Computer Specifications

Chapter 3

Implementation

What you actually did

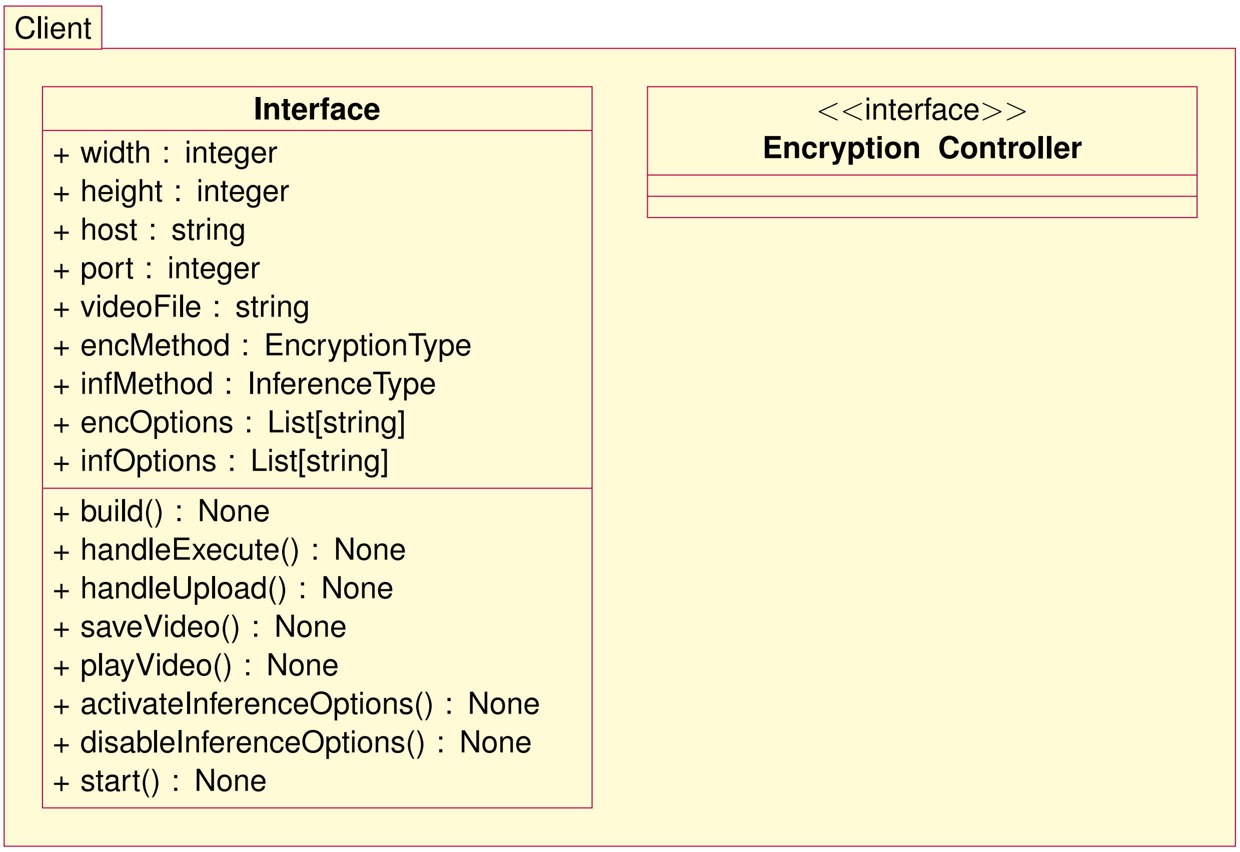


Figure 3.1: UML Class Diagram showing the components of the software.

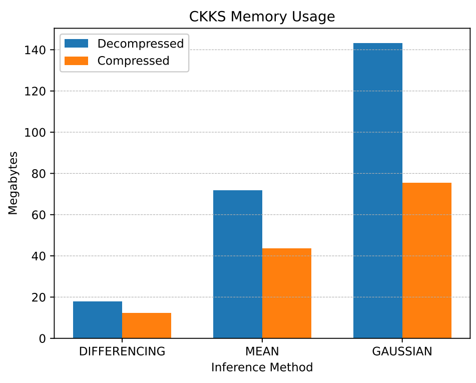
3.1 Implementation

3.2 Evaluation

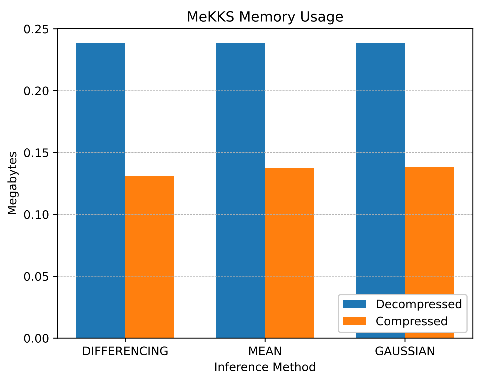
Chapter 4

Evaluation

4.1 Results and Analyses



(a) CKKS Encryption Scheme



(b) MeKKS Encryption Scheme

Figure 4.1: Memory Usage of Encrypted Frames

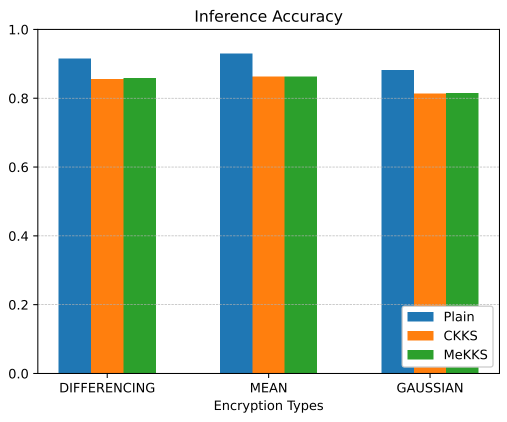
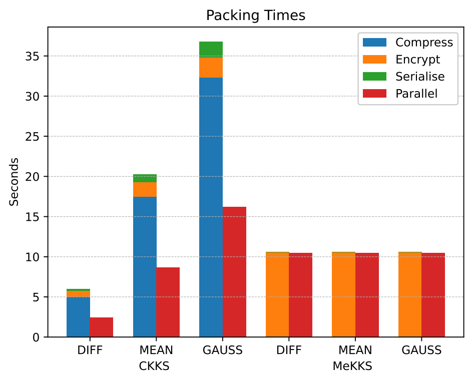
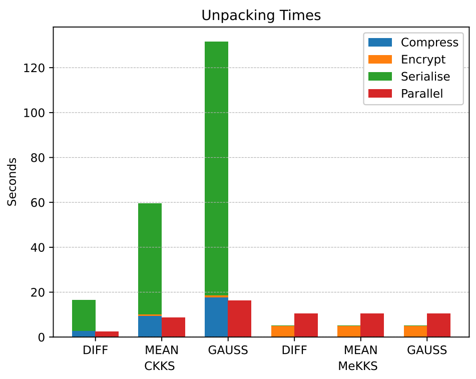


Figure 4.2: Inference Accuracy by Encryption Type

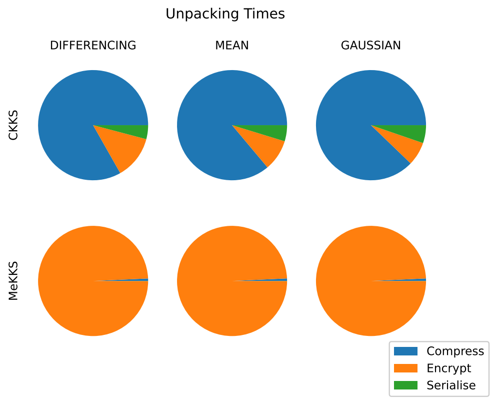


(a) Packing

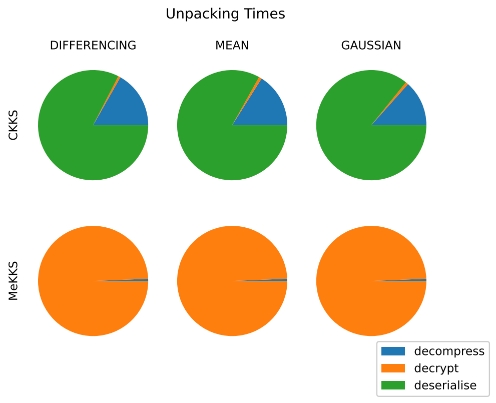


(b) Unpacking

Figure 4.3: Packing and Unpacking Times



(a) Packing



(b) Unpacking

Figure 4.4: Operations Performed During Packing and Unpacking

4.2 Discussion

Did your findings support your hypothesis? Why? Why not?

Chapter 5

Conclusions

This Chapter concludes the thesis by summarizing the findings from the study, the contributions and possible limitations of the approach. It can also identify issues that were not solved, or new problems that came up during the work, and suggests possible directions going forward.

Bibliography

[1] https://homomorphicencryption.org/standard/. Retreived March 2022.

[2] Ho Bae, Jaehee Jang, Dahuin Jung, Hyemi Jang, Heonseok Ha, Hyungyu Lee, and Sungroh Yoon. "Security and Privacy Issues in Deep Learning". In: *Journal of IEEE Transactions on Arificial Intelligence* (2018).

[3] M. Bhattacharya, R. Creutzburg, and J. Astola. "Some historical notes on Number Theoretic transform". In: *International TICSP Workshop on Spectral Methods and Multirate Signal Processing* (2004).

[4] Zvika Brakerski. "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP". In: *Annual Cryptology Conference* (2012).

[5] Brilliant.org. *Homomorphic Encryption*. https://brilliant.org/wiki/homomorphic-encryption/. Retrieved March 2022.

[6] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. "Bootstrapping for Approximate Homomorphic Encryption". In: *Advances in Cryptology* (2018).

[7] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: *Lecture Notes in Computer Science* (2016).

[8] Kuan-Yu Chu, Yin-Hsi Kuo, and Winston H. Hsu. "Real-time privacy-preserving moving object detection in the cloud". In: *Proceedings of the 21st ACM international conference on Multimedia* (2013).

[9] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy". In: *ICML* (2016).

[10] Eufy. *Eufy Homepage*. https://uk.eufylife.com/. est. 2016.

[11] Junfeng Fan and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption". In: *Cryptology ePrint Archive* (2012).

[12] Python Software Foundation. *PSF LICENSE AGREEMENT FOR PYTHON 3.10.4*. https://docs.python.org/3/license.html#psf-license. Retrieved March 2022.

[13] Python Software Foundation. *Python*. https://www.python.org/. Retrieved March 2022.

[14] Git. *Git*. https://git-scm.com/. Retrieved March 2022.

[15] Thore Graepel, Kristin Lauter, and Michael Naehrig. "ML Confidential: Machine Learning on Encrypted Data". In: *The International Conference on Information Security and Cryptology* (2012).

[16] Dazhou Guo, Yanting Pei, Kang Zheng, Hongkai Yu, Yuhang Lu, and Song Wang. "Degraded Image Semantic Segmentation With Dense-Gram Networks". In: *IEEE Transactions on Geoscience and Remote Sensing* (2021).

[17] Huelse. *SEAL-Python*. https://github.com/Huelse/SEAL-Python. Retrieved March 2022.

[18] Grupo de Tratamiento de Imágenes. *Labeled and Annotated Sequences for Integral Evaluation of SegmenTation Algorithms*. https://www.gti.ssr.upm.es/data/lasiesta_database. Retrieved March 2022.

[19] Github Inc. *Github*. https://github.com/. Retrieved March 2022.

[20] Open Source Initiative. *The 3-Clause BSD License*. https://opensource.org/licenses/BSD-3-Clause. Retrieved March 2022.

[21] Open Source Initiative. *The MIT License*. https://opensource.org/licenses/MIT. Retrieved March 2022.

[22] Sangpyo Kim, Wonkyung Jung, Jaiyoung Park, and Jung Ho Ahn. "Accelerating Number Theoretic Transformations for Bootstrappable Homomorphic Encryption on GPUs". In: *2020 IEEE International Symposium on Workload Characterization* (2020).

[23] Jaya S. Kulchandani and Kruti J. Dangarwala. "Moving object detection: Review of recent research trends". In: *2015 International Conference on Pervasive Computing (ICPC)* (2015).

[24] Kim Laine. "Simple Encrypted Arithmetic Library 2.3.1". In: *Microsoft Research TechReport* (2017).

[25] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST Database of Handwritten Digits*. http://yann.lecun.com/exdb/mnist/. Retrieved March 2022.

[26] Chih-Yang Lin, Kahlil Muchtar, Jia-Ying Lin, Yu-Hsien Sung, and Chia-Hung Yeh. "Moving object detection in the encrypted domain". In: *Multimedia Tools and Applications* (2017).

[27] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *Journal of the ACM* (2010).

[28] Microsoft. *OneDrive Personal Cloud Storage*. https://www.microsoft.com/en-gb/microsoft-365/onedrive/online-cloud-storage. Retrieved March 2022.

[29] Microsoft. *Visual Studio Code*. https://code.visualstudio.com/. Retrieved March 2022.

[30] Payal V. Parmar, Shraddha B. Padhar, Shafika N. Patel, Niyatee I. Bhatt, and Rutvij H. Jhaveri. "Survey of Various Homomorphic Encryption Algorithms and Schemes". In: *International Journal of Computer Applications* (2014).

[31] Manas A. Pathak and Bhiksha Raj. "Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models". In: *IEEE* (2013).

[32] Ring. *Ring Homepage*. https://en-uk.ring.com/. est. 2012.

[33] W. W. Royce. "Managing the development of large software systems: concepts and techniques". In: *Proceedings of the 9th International Conference on Software Engineering* (1987).

[34] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Pearson, 2001.

[35] Chris Stauffer and W. E. L. Grimson. “Adaptive background mixture models for real-time tracking”. In: *1999 IEEE computer society conference on computer vision and pattern recognition* (1999).

[36] Steven Turner. “Transport Layer Security”. In: *IEEE Internet Security* (2014).

[37] Ioannis Tzemos, Apostolos P. Fournaris, and Nicolas Sklavos. “Security and Efficiency Analysis of One Time Password Techniques”. In: *20th Pan-Hellenic Conference on Informatics* (2016).

[38] *Unsupervised Learning of Video Representations using LSTMs*. http://www.cs.toronto.edu/ nitish/unsupervised_video/. Retrieved March 2022.

[39] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland. “Pfinder: real-time tracking of the human body”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1997).

[40] Jingru Yi, Pengxiang Wu, Menglin Jiang, Qiaoying Huang, Daniel J. Hoeppner, and Dimitris N. Metaxas. “Attentive neural cell instance segmentation”. In: *Medical Image Analysis* (2019).

Appendix A

Project Proposal