This chapter discusses the preparatory work done before beginning the project’s implementation. §\ref{sec:preliminaries} introduces useful preliminary information and background, §\ref{sec:projectStrategy} discusses the methodologies used when approaching the design of the project’s implementation, and §\ref{sec:startingPoint} provides an overview of the foundations of the project.

This section is dedicated to introducing the fundamental underlying concepts, terminologies, and notations fundamental to this project. The project spans three domains of computer science: cryptography, graphics, 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 in places where further explanation is required.

# 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 access the data while it is stored by the server, exposing a range of privacy risks including identity theft, monitoring intimate behaviours of household members, observations of household contents, and more. Figure \ref{fig:threatModel} illustrates this succinctly. The first threat can be mitigated easily enough using cryptographic protocols such as TLS [TLS]. However, the second threat is much more difficult to defend against, particularly because data must usually be decrypted for inference to be performed [BAE].

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.

# 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 $\Pi = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$. \texttt{KeyGen} is a function used to generate a \textit{public key} (\texttt{PK}) and \textit{private key}\footnote{The \textit{private key} is referred to as a \textit{secret key} by some literature. While these terms are equivalent, general convention is to use \textit{secret key} in relation to symmetric encryption, and \textit{private key} when discussing asymmetric – a practice that this dissertation will follow.} (\texttt{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\footnote{An $l$-bit security parameter would require an expected $2^l$ attempts to guess the keys.}. 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 $\Pi$ into a HE scheme, a fourth function $\texttt{Eval}(f, c\_1, \ldots, c\_n)$ must be introduced. The evaluation function, $\texttt{Eval}$ applies a Boolean circuit, $f$, to the ciphertext arguments, $c1, \ldots, c\_n$ such that, for all arguments, it holds that

Equation

where $m\_1, \ldots, m\_n$ are the plaintext equivalents of $c\_1, \ldots, c\_n$. This is, perhaps, better illustrated by Figure \ref{fig:homomorphicEncryption} below.

Theoretically, a \textit{fully} homomorphic scheme\footnote{The prefix \textit{fully} derives from the existence of \textit{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 [PARTIALSCHEMES].} allows the evaluation of Boolean circuits indefinitely. However, in practice, the time complexity of operations means many schemes are \textit{levelled} homomorphic schemes. This means they only support operations up to a \textit{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 \textit{perfectly secure}, a ciphertext must provide no additional information about its plaintext (see Equation \ref{eq:def1}). In other words, the probability of generating a given ciphertext from a particular plaintext is independent of the plaintext (see Equation \ref{eq:def2}). The two equations below can be shown to be equivalent using Bayes’ rule.

Equations

While it is possible to create perfectly secure encryption schemes, they are impractical in real applications\footnote{For example, the One-Time Pad[OTP]}. Therefore, \textit{computational security} is considered sufficient. Relying on the hardness of certain mathematical problems, computational security means that an encryption scheme is \textit{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 rather than perfect security. More specifically, they utilise the hardness of the \textit{ring learning with errors} (henceforth RLWE) problem introduced by Lyubashevsky et al.\ [RLWE]. A polynomial-time reduction from the \textit{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, \textit{rings}. To understand \textit{rings}, \textit{groups} must first be understood. A group $(\mathbb{G}, \bullet)$ is a set, $\mathbb{G}$, and an operator, $\bullet: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$, such that the following properties hold:

\begin{itemize}

\item \textbf{Closure}: $a \bullet b \in \mathbb{G}$ for all $a,b \in \mathbb{G}$.

\item \textbf{Associativity}: $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all $a, b, c \in \mathbb{G}$.

\item \textbf{Neutral Element}: there exists an $e \in \mathbb{G}$ such that for all $a \in \mathbb{G}$, $a \bullet e = e \bullet a = a$.

\item \textbf{Inverse Element}: for each $a \in \mathbb{G}$ there exists some $b \in \mathbb{G}$ such that $a \bullet b = b \bullet a = e$.

\end{itemize}

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

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

\begin{itemize}

\item $(\textbf{R}, \boxplus)$ is an abelian group.

\item $(\textbf{R}, \boxtimes)$ is a monoid.

\item $\boxplus$ and $\boxtimes$ are distributive – for all $a, b, c \in \textbf{R}$, $a \boxtimes (b \boxplus c) = (a \boxtimes b) \boxplus (a \boxtimes c)$ and $(a \boxplus b) \boxtimes c = (a \boxtimes c) \boxplus (b \boxtimes c)$.

\end{itemize}

If $a \boxtimes b = b \boxtimes a$ then it is a \textbf{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 $\mathbb{Z}\_q$\footnote{$\mathbb{Z}\_q$ is the set of integers modulo $q$. For example, $\mathbb{Z}\_7 = \{0, 1, 2, 3, 4, 5, 6\}$.}. Known as a \textit{quotient ring}, this can be denoted by $\mathcal{R}\_q = \mathbb{Z}[X] / (\Phi(X))$, where $\Phi(X)$ is an \textit{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, $\vec{v}$, as a list of polynomials and then using a secret polynomial to convert this list to a polynomial, it is infeasible to recover $\vec{v}$ in polynomial time.

The HE schemes discussed in this dissertation rely on the RLWE problem to assert \textit{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 \textit{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.

Practically, HE schemes choose a \textit{cyclotonic polynomial} for $\Phi(X)$, where the $n$-th cyclotonic polynomial, $\Phi\_n(X)$, is defined as

Equation

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 \textit{number theoretic transform} (NTT)\footnote{A specialisation of the discrete Fourier transform, the NTT is a generalisation of the Fast Fourier transform in the case of finite fields [NTT].}. The advantage of this is that it can be easily accelerated using hardware [HARDWARE].

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 $q$ [HEStandard]. Therefore, a fundamental trade-off is introduced between the supported depth of multiplication and the security level.

# Moving Object Detection

## Introduction

A discussion of \textit{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 [SHAPIRO].

There are two broad categories of segmentation techniques: \textit{semantic segmentation} and \textit{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’’ [SEMANTIC]. 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 [INSTANCE].

One application of image segmentation is \textit{foreground extraction}, also known as \textit{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 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, \textit{frame differencing} works by iterating through the frames of a video in a single pass [FRAMEDIFFERENCING]. 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\footnote{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}. 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 \ref{eq:frameDifferencing}.

Equation

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

## Mean Filter

A \textit{mean filter} approach to moving object detection attempts to overcome the weaknesses of selecting a reference frame when performing frame differencing [MEANFILTER]. Instead of taking a frame directly from the video, the value of $B$ at time $t$ is calculated using Equation \ref{eq:meanFilter}.

Equation

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 \ref{eq:frameDifferencing}.

## Median Filter

Performing moving object detection using a \textit{median filter} is almost identical to the mean filter method [MEANFILTER]. 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 \ref{eq:frameDifferencing}.

## Gaussian Average

Wren et al.\ originally proposed fitting a Gaussian probabilistic density function to the most recent $N$ frames [WREN]. 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 \ref{eq:mean}, and the variance using Equation \ref{eq:variance}.

Equation

Equation

where $\alpha$ determines the size of the \textit{temporal window} used to fit the Gaussian model\footnote{$\alpha$ 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’.}, $d = |f\_t - \mu\_t|$ 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 \ref{eq:meanThreshold}

Equation

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 \textit{Gaussian mixture models} (henceforth GMMs) for moving object detection in 1999 [STAUFFER]. GMMs are probabilistic models that represent 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 \textit{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 \textit{component weights}, and the component \textit{means} and \textit{variances}. For a GMM with $N$ components, the $i^\text{th}$ component has $\mu\_i$ and variance $\sigma\_i$ in the \textit{univariate case}, and mean $\vec{\mu}\_i$ and a covariance matrix $\Sigma\_i$ in the \textit{multivariate case}. The component weights, $\phi\_k$ for component $k$, are constrained by the equation $\sum^K\_{i=1} \phi\_i = 1$. If the component weights aren’t learned, they are known as an \textit{a-priori}\footnote{A probability derived purely through deductive reasoning.} distribution over components such that $\probP(x \text{ generated by component k}) = \phi\_k$. If the component weights are learned, they are known as \textit{a-posteriori}\footnote{From Bayesian statistics, referring to conditional probability $\probP(A | B)$.} estimates of the component probabilities given the data.

There are several methods for fitting the GMM’s parameters. A common method is to use \textit{expectation maximisation} (EM), if the number of components of the GMM is known. When fitting a GMM, the Gaussian distributions are being tuned to match the distributions observed in the data. If all of the data is known, it can all be incorporated into this stage to achieve the most accurate fitting. However, in practice, it is likely that only a subset of the data will be available, so the GMM will then have to extrapolate to the real values. This technique can also be taken advantage of if there is limited time in which to perform fitting.

The EM algorithm is split into two many stages. Firstly, the E-step calculates the expectation of assigning each data point to each component of the GMM, given the GMM’s parameters. Secondly, the 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\footnote{The E-step corresponds to the latter case, and the M-step corresponds to the former.}. Therefore, by alternating which values are assumed known, maximum-likelihood estimates of the unknown values can be efficiently calculated.

Once a GMM has been fitted, it can be used for inference. There are two common uses of inference: \textit{density estimation} and \textit{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.