# Knowledge and Experience

Prior to beginning the project, the following relevant Tripos courses had been completed: \textit{Scientific Computing}, \textit{Machine Learning and Real-world Data}, \textit{Software and Security Engineering}, \textit{Concurrent and Distributed Systems}, \textit{Data Science}, \textit{Computer Networking}, and \textit{Security}. The Part II course, \textit{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 theory was learned independently of Tripos studies. The study of applied HE is sparsely documented, so most understanding came from academic papers; notably [CKKS] and [SEAL]. Although, articles such as [BRILLIANTHE] were more useful for foundational knowledge.

Similarly, there was little mention of computer vision artificial intelligence in Tripos, so most understanding came from independent research. Academic papers such as [STAUFFER] and [KULCHANDANI] 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.

# Tools Used

## Programming Languages

All code written for this dissertation was written in \textit{Python} [PYTHON]. The main reasons for this were the large machine learning ecosystem, ease of use, and ease of debugging. These factors allowed for quick implementation, making Python best suited to the project’s tight schedule.

However, Python is not a language traditionally used for cryptographic applications. Usually, lower-level, faster languages like C++ are favoured. Since the 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

\textit{Visual Studio Code} [VSCODE] 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, \textit{Git} [GIT] and \textit{GitHub} [GITHUB] were used for version control and source code management. \textit{OneDrive} [ONEDRIVE] 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 availability of academic literature discussing them. In particular, the CKKS scheme [CKKS] was selected because it supports representing real numbers\footnote{Versus the BFV scheme, which only supports integers [BFV].}. However, the project is designed to allow any HE scheme following the same API to be substituted in CKKS’s place.

## Libraries

The project uses Microsoft’s SEAL library [SEAL], 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. SEAL was integrated using a Python wrapper library [WRAPPER].

## Datasets

There were two publicly available datasets used in this project:

* The Moving-MNIST dataset contains ten thousand sequences of length twenty frames showing two handwritten digits from the standard MNIST dataset moving in a $64 \times 64$ 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. [MOVINGMNIST, MNIST].
* The LASIESTA dataset contains sequences showing individuals moving across static backgrounds. Specifically designed to evaluate segmentation algorithms, this dataset provides more realistic examples of surveillance footage so allows a truer evaluation of moving object detection in the HE domain. [LASIESTA].

## 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 \ref{tab:licensing} gives the specific licenses.

NumPy – 3-clause BSD

SEAL – MIT

SEAL-Python – MIT

Python – PSFL

Matplotlib – PSFL

Tkinter - PSFL

Multiprocessing – 3-clause BSD

Sympy – 3-clause BSD

PyJoules -MIT

# 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 \ref{tab:specs}.