**City, University of London**

**MSc in Cyber Security**

**Project Report**

**2023-2024**

# Zero Knowledge Proofs for new reCAPTCHA

**Katerina Giagkoumpova**

**Supervisor: Nikos Komninos**

Date of submission: 16th of October 2024

## Declaration

*By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.*

*Signed: Katerina Giagkoumpova*

## Abstract

reCAPTCHA v3, Google’s latest verification system, can filter humans from illegitimate bots without user interaction by using a risk analysis engine gathering the web client’s background data. Despite how widespread this system has become, web clients have become suspect of whether their data is private, and it has been criticised to being susceptible to bots, with literature showing that machine learning techniques such as Reinforcement Learning can achieve up to 99.6% evasion rates. With machine learning continuing to advance, these weaknesses underscore the need for a more secure CAPTCHA solution. This project investigates a possible solution by integrating a Zero Knowledge Proof protocol onto the domain of a One-Class Support Vector Machine (SVM). This would allow the machine learning model to operate without revealing internal mechanisms nor the web client’s data being used, ensuring both verification of the model but also privacy preservation. This project lays the groundwork for a potential reCAPTCHA v4, which uses a client’s behavioural data while at the same time preserving their privacy.

Keywords: reCAPTCHA, machine learning, zero knowledge, zkSNARK, One-Class SVM

## Acknowledgments

I would like to express my gratitude to my supervisor, Nikos Komninos, for his guidance and support throughout the duration of the project. I would also like to thank my family and friends for their encouragement and support.

## Contents

# 1 Introduction

## 1.1 Background

Illegitimate bots have long been a challenge for web hosts capable of malicious acts such as service disruption through Distributed Denial of Service attacks. It is therefore crucial for website hosts to distinguish between human web clients and bots. Google's reCAPTCHA verification system has been instrumental in this area.

CAPTCHA has evolved over the years, with the latest iteration being reCAPTCHA v3. Past versions required users to solve challenges to verify they were human, meanwhile reCAPTCHA v3 is frictionless, with minimal interaction necessary from the web clients. According to (Google Developers, 2018), this version utilises adaptive risk analysis, potentially indicating the use of a combination of machine learning (ML) algorithms.

This version is user-friendly as its' backend server analyses the web client's behavioural characteristics and determines a score between 0 to 1.0, with 0 denoting high likelihood of bot activity and 1.0 high likelihood of human activity. This method, however, is not foolproof. Akrout et al., (2020) provide the first research paper using reinforcement learning models to act like humans on the web, evading reCAPTCHA with more than a 90% evasion rate. Tsingenopoulos et al., (2022), corroborates this with a similar experiment ultimately achieving a 99.6% evasion rate of reCAPTCHA. Kotaro's, (2022) Masters thesis, evaluated the system for vulnerabilities, discovering four properties of v3, specifically,

1. The default score that the system produces is 0.9 out of 1.0 with deviations from the data that is expected, decreasing it.
2. For more than 500 consecutive requests to connect, the web crawlers become identified as bots only on the subsequent day.
3. A sleep duration of ten seconds between requests does not flag web crawlers as bots.
4. Requests with matching language codes in the "accept-language" header and the IP address location, and dynamic user behaviour, such as keyboard typing, could possibly evade reCAPTCHA v3's bot detection mechanism.

While these highlight the inability of reCAPTCHA v3 to detect bots, these issues are persistent due to Google not publishing the model for the verification of its claimed results and analysis for feedback by third parties.

Another crucial issue that Google faces are concerns from web users regarding data privacy. According to Dinh and Hoang, (2023), this concern arises since the client's behaviour and sensor data gets transmitted to remote servers. The paper mentions a suggested solution where only the decision score gets transmitted to the server; however, this approach needs trusted hardware to ensure privacy from the client’s side. Consent Management Platform (CMP), (2024) states that reCAPTCHA v3 is not GDPR-compliant and in order to become compliant, requires extra steps to be taken by the web developer and recommends the use of v2 instead, for better privacy.

Given the widespread use of reCAPTCHA, all the weaknesses that have been presented, pose a significant risk and prompt the need for robust solutions. Zhang et al., (2020) points to the overall issues regarding ML models, emphasising the lack of transparency in the accuracy and claimed results of models. As the training data and the algorithms cannot be published for security protection, their accuracy cannot be verified by third parties. An example of this is in Grant, (2020) when a company falsely claimed its’ food delivery robots were autonomous when they were controlled by humans.

The same paper from Zhang et al., (2020), proposes that by integrating a Zero-Knowledge Proof (ZKP) protocol into the domain of a Machine Learning (ML) model, allows to verify the claimed integrity of the model without compromising its privacy. This means that the model can run as intended without leaking any information about how it actually works. This idea is an existing application in the research space noted as, Zero Knowledge Proofs for Machine Learning (ZKML), with vast amounts of applications in different fields. This project proposes a ZKML framework that can lead to a secure Google reCAPTCHA v4 verification system.

In summary, Google’s reCAPTCHA v3 system has been a key player for filtering the bots from the human web users for the past couple of years. This service, by potentially using machine learning becomes prone to integrity issues as is the case with many other ML algorithms on the market, with studies showing up to 99.6% evasion rate of reCAPTCHA v3, ultimately failing to detect all bots. These findings, combined with the concerns over the lack of transparencies in ML algorithms, which reCAPTCHA most likely uses, underscore the need for more robust and verifiable solutions, bringing into question whether a ZKML framework can aid in this.

## 1.2 Output

The output of this project is a ZKML that merges ZKP with ML to provide a more secure and transparent verification process, reCAPTCHA v4.

## 1.3 Aims and Objectives

The aim of this project is to explore how non-interactive Zero-Knowledge Proofs (ZKP) can be integrated into the domain of ML models to patch the vulnerabilities of reCAPTCHA v3.

The following presents the objectives of the project and how they are met:

**Objective 1:** Identify what characteristics determine human web browsing behaviour.

This objective is achieved through research and is supported by the dataset used.

**Objective 2:** Identify which parameters of non-interactive ZKPs can be implemented into CAPTCHA.

Two ZKP protocols were researched and evaluated based on their properties and how they are compared in past literature. The most suitable was the one that fit to the ZKML's specification requirements and later on was adapted to suit the specific scenario.

**Objective 3:** Identify which characteristics of ML algorithms can be implemented into CAPTCHA.

This objective was met by researching three main ML algorithms and examining which is most compatible with the chosen dataset.

**Objective 4:** Evaluate how effective the integration of ZKPs and ML models (ZKML) is when implemented into CAPTCHA.

This objective was achieved by conducting an evaluation of the proposed ZKML's design, its' partial implementation and its' benefits and drawbacks. The ZKML's position in literature was explored, and its' security was assessed through the creation of a potential threat model.

## 1.4 Beneficiaries

- Google

As reCAPTCHA is Google's product, the company will benefit the most. This project will aid in protecting their customers and their personal data by simultaneously allowing for transparency and preserving the integrity of CAPTCHA. As has been cited in 1.1, the service is not GDPR compliant hence, this addition of a ZKP will make the service more trusted to the web users, improving its' reputation.

- Web users

With over 3 million customers worldwide, reCAPTCHA is the most widely used captcha verification service (6sense, 2024). Bots may perform numerous malicious activities on websites resulting in stolen customer data including credit card and address information. Therefore, increased protection can prevent and support in the safeguarding of customer data.

- Website hosts

Website owners trust Google's service with many of them depending on its accuracy as they may not have adequate cybersecurity training or funding for an expert team. By reCAPTCHA doing its job as successfully as possible with as few bypasses as possible, web hosts are able to have a safer website.

- Researchers

The output of the project will add on to the extensive literature of ZKML applications and provide a foundational framework for its' application in reCAPTCHA.

## 1.5 Research Question

How can zero-knowledge proofs aid in the development of a secure reCAPTCHA v4?

## 1.6 Reasons for Choosing This Project

Initially sounding like an interesting idea, further research for the project proposal showed that ZKML has not yet been applied to reCAPTCHA, making this project intriguing to pursue. The available literature is relatively new and emerging, giving the author the chance to expand their knowledge. To add, the author viewed this project as an opportunity to enhance their research skills and take on a challenge with such a complex background.

## 1.7 Outline of Methods

The method to complete this project include conducting thorough research on each separate topic, that being Google's CAPTCHA, Machine Learning algorithms, Zero-Knowledge Proofs and their integration, namely, Zero-Knowledge Proofs for Machine Learning (ZKML). Once this is done, the framework is designed and later on partially implemented using the Python programming language. The code is then tested on unseen datasets and the overall framework is evaluated.

## 1.8 Major changes

A major change that occurred was that in the design stage, the dataset used was not examined properly, which lead to some changes required. The issue was that the ML algorithm initially chosen, the K-Nearest Neighbour (KNN), requires two classes to compare the testing data to, however, the dataset only contains one class that defines human web browsing activity. According to (Li et al., 2003), one class SVM can be used as a means for bot detection. Hence, this project implements a one-class SVM model. Due to the dataset containing only human defining data and no bot data, the model cannot definitively state that if the user is not found to be human then they are definitely a bot.

## 1.9 Structure of report

Chapter 2 Context: The necessary context of the subjects explored in this project.

Chapter 3 Methods: The description of the steps that were taken to meet the objectives.

Chapter 4 Results: The results of the project.

Chapter 5 Discussion: A discussion of whether the objectives have been met.

Chapter 6 Evaluation, Reflections, Conclusion: An evaluation of the project overall, limitations, future work, a reflection of the project conducted and an overall conclusion summarising what was done.

The appendices:

Appendix A: The project proposal submitted in May 2024 based on this project.

Appendix B: The source code of the implementation on user 4, 9 and 10.

Appendix C: The source code of the partial implementation for users 0, 2, 3, 5, 6, and 8.

Appendix D: Notes from Meetings 1-9.

# 2 Context

## 2.1 Google's CAPTCHA

### 2.1.1 CAPTCHA Overview

CAPTCHA is an acronym for 'Completely Automated Public Turing' test to Tell Computers and Humans Apart and as the name suggests is a test that distinguishes between humans and computers – bots. It is used in numerous applications such as online voting, email registration, search engines and to prevent distributed denial of service according to Xu et al., (2020).

reCAPTCHA is a CAPTCHA system developed by Google Inc. consisting of three versions, version 2 and 3 currently in use. Version 1 displayed distorted words or letters, asking the user to identify them, as seen in Figure 1. It used Optical Character Recognition (OCR) (Eikvil, 1993) and worked on the assumption that humans have a visual memory of these letters unlike a computer. However, as bots improved, text distortion could be bypassed a lot easier hence, it is now discontinued (Google for Developers, 2018).

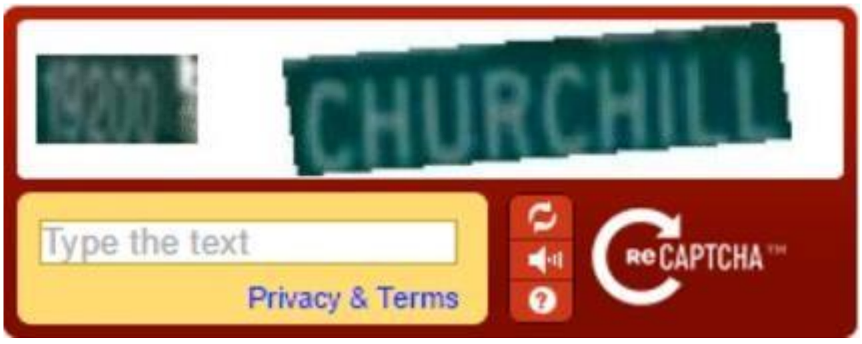


*Figure 1: Example of reCAPTCHA version 1 from (Jiang, et. al., 2017)*

In 2013, Google introduced a different reCAPTCHA system, the "No CAPTCHA reCAPTCHA", requiring users to either click a checkbox or choose images by identifying certain objects for mobile devices. This is seen in Figure 2. This version also utilises OCR technology, that according to Sukhani et al. (2021) can be bypassed using Convolutional Neural Networks.

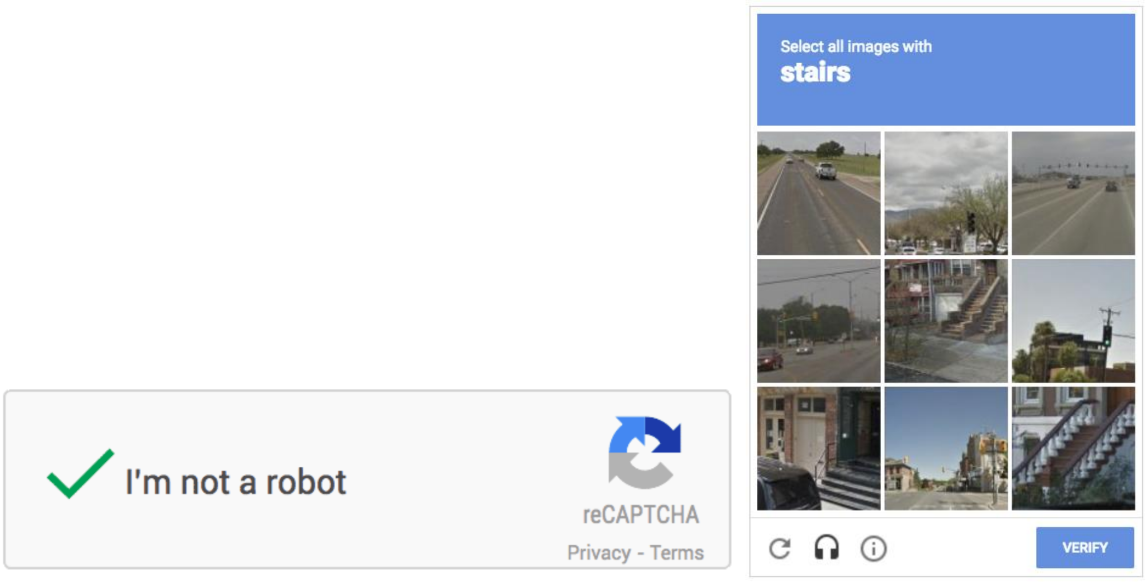


*Figure 2: reCAPTCHA v2 examples (Left: (Swisher, 2024) Right: (Ritthichai, 2020))*

### 2.1.2 Google's reCAPTCHA v3

In 2017, the latest instalment of Google's renowned CAPTCHA system was published, reCAPTCHA v3. Unlike version 2, this version is a completely "frictionless" experience for the user as it does not require them to click on anything. Instead, it leverages adaptive risk analysis to analyse user behaviour in the background. The interactions it monitors include mouse clicks, scrolling patterns and the how long a user stays on a webpage among other data that describes a user's behaviour (Ghimire, 2023). It then derives a risk score, ranging from 0 to 1.0, with higher scores indicating a higher likelihood of bot activity. Google gives the website owners the possibility of defining a score threshold to determine the level of risk that may be associated with a user. Sometimes, for scores below the threshold, they may be a requirement for additional verification steps, while score above the threshold allow access to the website.

This lack of interaction from user makes it user-friendly, meanwhile the dynamic scoring system permits the website owners to adapt the threshold to their own preferences based on their own website needs and requirements.

The following describes how the client-server interaction works for reCAPTCHA v3.

As seen in Figure 3, the main components of the client-server interaction are the website, the web browser's backend server and the reCAPTCHA v3 backend server. The following steps describe how it works and is in reference to Raychev, (2022).

- The user sends an HTTP request to the website which uses the site's public key to send a request to the reCAPTCHA's backend API.

- A response token is generated on the backend that is sent back to the website. This response token is valid for only 2 minutes and cannot be reused.
- The token on the website is then forwarded to the backend web server which in turn sends the token as well as the secret key to reCAPTCHA's backend.
- The backend reCAPTCHA server generates a score between 0 to 1.0.
- This score is then sent to the backend of the web browser that will eventually determine whether the user is indicating a human or an illegitimate malicious bot based on the web hosts' settings.
- This result is then sent to the site which in turn accepts or rejects the user from entering the website.

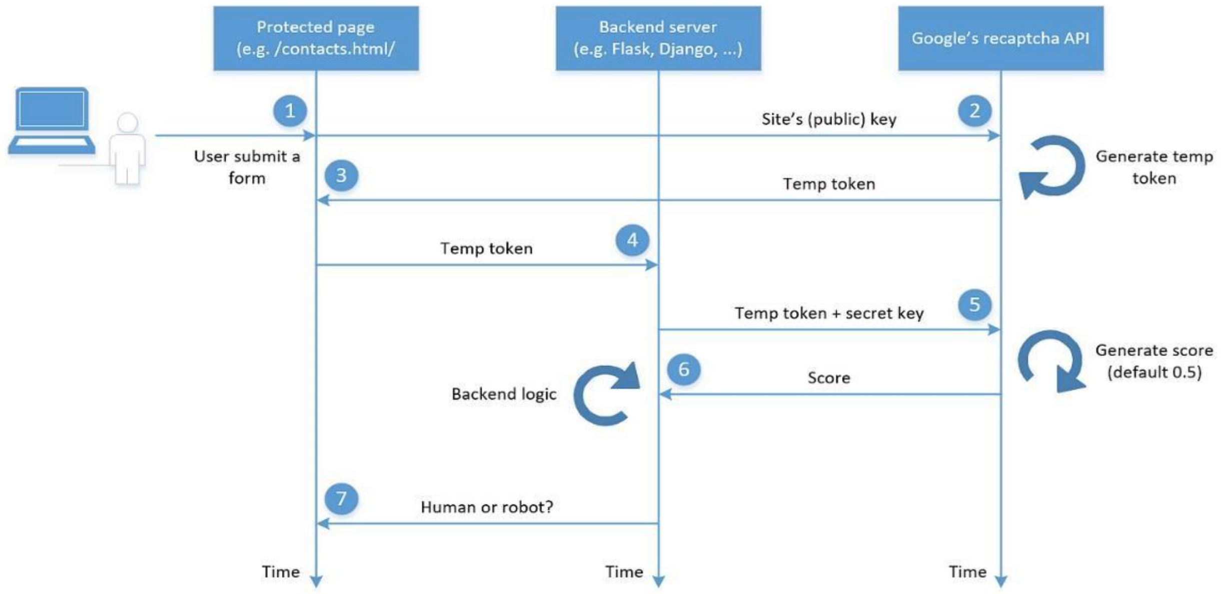


*Figure 3: Client - Server Interaction (Raychev, 2022)*

## 2.2 Machine Learning

Google for Developers, (2018) states that reCAPTCHA version 3 uses adaptive risk analysis, an indication of the potential use of Machine Learning (ML) algorithms.

Machine Learning is a computational approach where a problem can be solved through an algorithm by learning from past examples (Jo, 2021). There are four main categories of ML algorithms: Supervised, Unsupervised and Semi-Supervised and finally, Reinforcement Learning.

Each category is mainly defined by the type of data it is trained and tested on. This project explores algorithms of the supervised learning class namely, KNN, RNN as well as one algorithm of the unsupervised learning class, one-class SVM.

### 2.2.1 Supervised learning algorithms

Supervised algorithms are defined by the inputs they take and outputs they produce. More specifically, the data is labelled, meaning every input the model is trained on has a corresponding output. Hence, the new data that will be put into the model, the data it has not seen yet, are classified into an output based on the trained data (Jo, 2021). Some algorithms that are explored are the KNN, RNN and SVM.

The KNN algorithm operates by comparing the similarity of the new, unseen data with the training examples using different metrics like Euclidean, Manhattan or Minkowski distance (Jo, 2021). Both KNN and RNN classify new input data into one of the two classes it is trained on.

The SVM algorithm is a dual hyperplane classifier aiming to achieve the maximum margin in a mapped space known as feature space (Jo, 2021). In the feature space, the similarity between two vectors is calculated using a kernel function. This kernel function is defined "as the inner product of two vectors in the feature space" (Jo, 2021). There are different types of kernels such as the $linear$, mapping a linear decision boundary, the radial basis function $rbf$, mapping to an infinite-dimensional space ideal for complex patterns, and the polynomial $poly$, mapping the input features into a polynomial space (Jo, 2021). Some other hyperparameters included are the $\gamma$ value and the $\nu$ (Shrestha, 2024).

### 2.2.2 Unsupervised learning algorithms

The unsupervised algorithms are characterised by the training data being unlabelled meaning that for the training data the input does not map to a specific output.

One-class SVM is an unsupervised algorithm that acts just like the supervised SVM with the main difference being that it classifies the input features into only one class. It identifies the boundary for normal data points and any data outside said boundary is classified as an anomaly (Kun-Lun Li et al., 2004). To find the hyperplane of the One-class SVM, the following optimisation problem finds the smallest hyperplane and is defined by Schölkopf et al., (1999) as such,

$$\min_{w\in F,\xi\in\mathbb{R}^l,\rho\in\mathbb{R}} \frac{1}{2}\left||w|\right|^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho$$

$$\text{subject to } \left(w \cdot \Phi(x_i)\right) \geq \rho - \xi_i, \quad \xi_i \geq 0$$

where $x_1, \dots, x_l$ is the training data, and $l \in \mathbb{N}$ the number of observations. $w$ is the normal vector defining the hyperplane, with $\Phi(x_i)$ the mapping function that transforms data into a higher-dimensional space. $\nu \in (0,1]$ is the parameter that controls the fraction of outliers allowed in the data and the $\rho$ is the margin.

The OC-SVM generates a decision function to identify the outliers for each new point $x$. According to Fragoso et al., (2016), if the function,

$$f(x) = sgn\{\left(\left(w, \Phi(x)\right) - \rho_1\right)\left(\rho_2 - \left(w, \Phi(x)\right)\right)\}$$

is positive, $x$ is classified as normal, and if it is negative then $x$ is considered an outlier.

## 2.3 Zero-Knowledge Proofs

### 2.3.1 Zero-Knowledge Proofs

The integration of ZKPs into the domain of the ML can address the issues present in ML algorithms (Zhang et. al 2020).

Zero-Knowledge Proofs (ZKP) are defined as the cryptographic protocols allowing one party, denoted as the prover, to showcase to another party, denoted as the verifier, that they possess knowledge of a secret without explicitly stating what the secret is. This way, the prover can demonstrate to the verifier that they know some information without saying what the information is. ZKP is a foundational building block for preserving privacy and confidentiality in the several fields including cryptography and blockchain technologies.

A ZKP is built on three properties (Sun et al., 2021):

**Completeness**: The verifier will always believe that the prover's statement is true.

**Soundness**: If the prover's statement is false then they cannot convince the verifier that this statement is actually true, the verifier will reject it.

**Zero-Knowledge**: the prover does not reveal any valuable information to the verifier, the verifier learns nothing about the statement besides the act that it is truthful.

A ZKP Framework contains the following three phases also visualised in Figure 4:

- "Witness Phase: The prover computes a proof that contains its statement. Then, the proof is transmitted to the verifier.
- Challenge Phase: The verifier asks the prover several questions.
- Response Phase: The prover answers these questions, which can be used for the verifier to accept or reject the generated proof." (Sun et al., 2021)

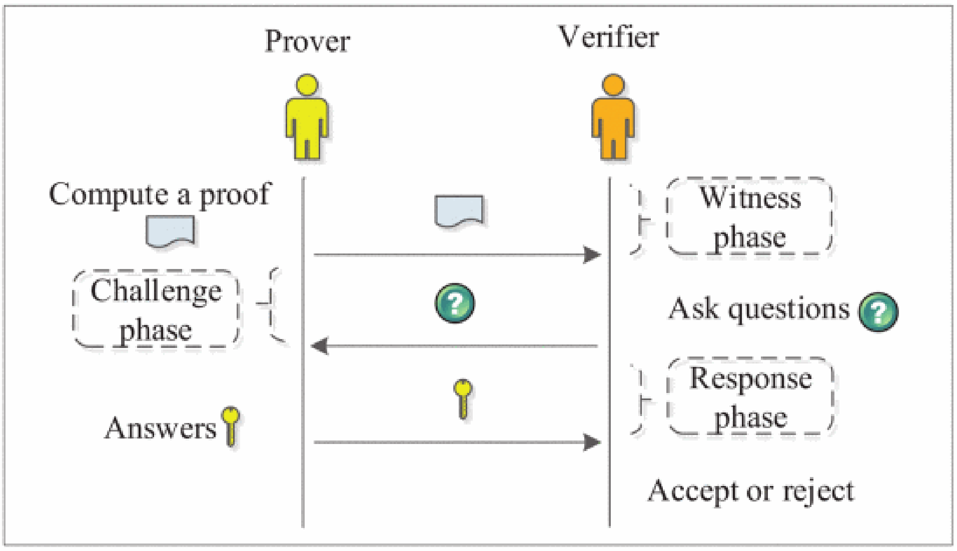


*Figure 4: ZKP framework (Sun et al., 2021)*

There are two types of ZKPs. The interactive ZKP and the non-interactive ZKPs. Interactive ZKP protocols require a number of interactions between the two parties, the prover and verifier, until the process is completed, and the verifier is convinced. On the other hand, non-interactive ZKP protocols allow the prover's proof to be verified in a single step (Mulder et al., 2023). This project focuses on two non-interactive ZKP protocols, namely, zkSNARK, and Bulletproofs.

### 2.3.2 zkSNARK protocol

zkSNARK stands for Zero-Knowledge Succinct Non-Interactive Argument of Knowledge. In this protocol, there is one initial setup phase usually done by a trusted authority, followed up by one message that completes the interaction between the two parties.

In addition to the three standard properties of a ZKP, it also contains two more properties. Firstly, it is succinct, so the proof size is always much smaller to the size of the computation. Secondly, it exhibits an "argument" property; where the outcome of any computation can serve as a statement or argument by the prover (Mulder et al., 2023).The zkSNARK protocol is comprised of three algorithms as described in (Nitulescu, 2019):

- $Gen$ – The setup algorithm which is usually conducted by a trusted authority. It produces the crs string which will be useful for creating the proving and verification keys in the following algorithms.
- $Prove$ – The proving algorithm conducted by the prover. It takes an input of the proving key $pk$, the statement $x$ and the witness $w$, with the product resulting in the proof $\pi$.
- $Verify$ – The verification algorithm conducted by the verifier. The input is the verification key $vk$, the statement $x$ and the proof $\pi$. The output is the decision of whether the statement has been proved as true or false.

Now, a circuit is necessary to be able to develop the zkSNARK. An arithmetic circuit is made up of wires that transmit values for a field F and connect to addition and multiplication gates (Nitulescu, 2019). This is seen in Figure 5 below. In this figure, a and b are added together with its sum being multiplied with value c. The output is 30. This circuit corresponds to $(a + b) \times c = 30$.

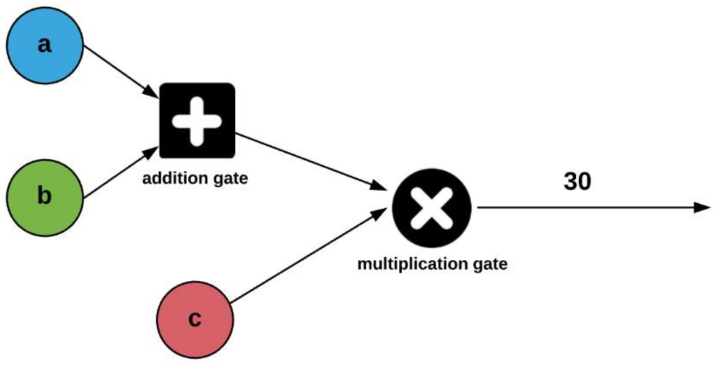


*Figure 5: Arithmetic circuit (Zeilberger, 2019)*

A cryptographic hash function, can be applied to the secret of the zkSNARK to protect the privacy of the secret data. A valid hash function must meet the following two properties:

- "One-way: It is computationally infeasible to find any input that maps to any pre—specified output.
- Collision Resistant: It is computationally infeasible to find any two distinct inputs that map to the same output." (Grassi, Garcia and Fenton, 2017)

### 2.3.3 Bulletproofs protocol

Bulletproofs are ZKP protocols that prove "a secret committed value lies in a given interval" (Bunz et al., 2018). Due to this, they are particularly useful in privacy preserving applications such as confidential transactions in blockchains and in online voting. In these systems, it is essential for users to be verified of the validity of their transactions without exposing any sensitive information.

A relatively new cryptographic technique, they are non-interactive and unlike other ZKP protocols, they avoid the requirement of a trusted setup (Morais et al., 2019). The construction of this protocol involves committing the secret value using a Pedersen commitment scheme (Pedersen, 1992) and utilising an inner product argument to prove that this secret value lies within the specific range. With the help of this inner product argument, the verifier can confirm whether the proof is valid (Bunz et al., 2018).

## 2.4 Zero-Knowledge for Machine Learning

The topic of integration of ZKPs into ML algorithms is well studied, with numerous implementations. This chapter explores some of the relevant literatures to this project and its' output.

In (Lee et al., 2024) the authors discuss a scenario where a hospital sends an AI doctor CT scans and the AI doctor diagnosed them on a disease. The point is to make it possible for third parties such as the patient themselves as well as an insurance company to verify the claimed accuracy of this AI doctor without having access to the weight values of the model. This is addressed using zkSNARK protocols which aids in verifying the result of the model without requiring the client to know the input data and weight values used. The paper discusses a new version of this, the "vCNN", that makes performance faster by proposing a new Quadratic Arithmetic Program (QAP) formula for convolution, and its' results show significant time reduction for example the setup time for the model VGG16 reduced from 13 years to 10 hours. However, (Feng et al., 2024) reduces proof time for VGG16 from 6 minutes to 48 seconds, making it more practical.

To add, the zkCNN system, allows the owner "to prove to others that the prediction of a data sample is indeed calculated by the model, without leaking any information about the model itself" (Liu et al., 2021). It can be used to prove how accurate a secret CNN model is on a public dataset.

Some other literature regarding this includes ZEN (Feng et al., 2021), DIZK (Wu et al., 2018), ZENO (Feng et al., 2024), Zero knowledge proofs for decision trees (Zhang et al., 2020), Mystique (Weng et al., 2021), ZKP-VML (Xing et al., 2023), ezDPS (Wang and Hoang, 2023), and VFL-Chain (Abla et al., 2024).

# 3 Methods

## 3.1 Overview

This project follows the waterfall method. To lead to the desired output, a thorough literature review is conducted to gain a deeper understanding of the subjects the project examines. Then, a low-level scenario is designed and once the most suitable ZKP is chosen, the scenario turns into high level as is displayed in Figure 6. Following this, a partial implementation of the ZKML is conducted and then tested on an unseen dataset. The framework is then evaluated.

This project is split into two main parts: the ZKML's circuit construction and the implementation of the verification of the system. The former is constructed theoretically whilst the latter is the proof of concept and established using the Python programming language.

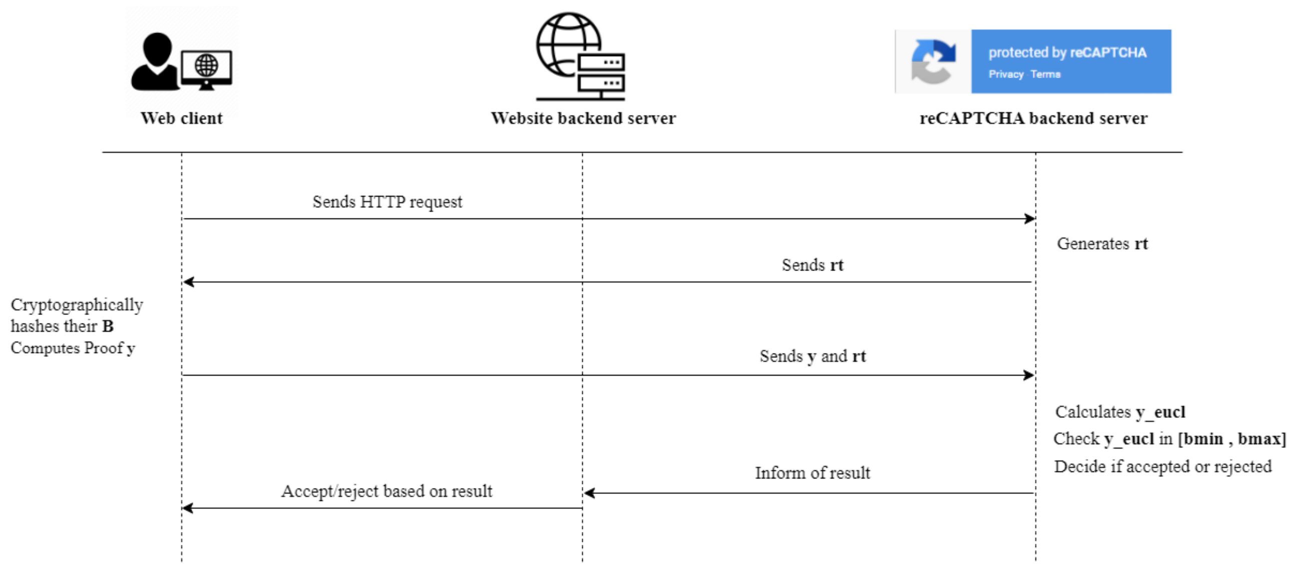


*Figure 6: ZKML Protocol Flow*

### 3.1.1 Notation

Below are the notations that the project uses mainly in the *Chapter 4 Results*.

*Table 1: Notations*

| Notation | Description |
|---|---|
| $\boldsymbol{TA}$ | The trusted authority responsible for the trusted setup stage and the setup algorithm. |
| $\boldsymbol{pk}$ | The proving key used in the proof generation algorithm. |
| $\boldsymbol{vk}$ | The verification key used in the verification algorithm. |
| $\boldsymbol{\lambda}$ | Security parameter |
| $\boldsymbol{C}$ | The zkSNARK's circuit |
| $\boldsymbol{rt}$ | Response token from the server to the client |

| | |
|---|---|
| $\vec{B}$ | The behavioural data of the client denoted in vector form. |
| $H$ | The cryptographic hash of the behavioural data. |
| $\vec{y}$ | The proof generated from the ZKP /the product of scalar c and vector B. |
| $x_i$ | The vectors of the training set with I denoting the number of features. |
| $y_i$ | The vectors of the proof. |
| $y_{eucl}$ or $y_{eucl_i}$ | The Euclidean distances between y and the training set. I denotes each distance 1,2,…,n. , n the number of features. |
| $bmin$ $bmax$ | The minimum and maximum values of the range of the model. |
| $Gen$ | The setup algorithm of the zkSNARK protocol |
| $Proof$ | The proving algorithm of the zkSNARK protocol |
| $Verif$ | The verification algorithm of the zkSNARK protocol |
| $m$ | The statement being proven. |

### 3.1.2 Assumptions

The project's assumptions are outlined below.

*Table 2: Assumptions of the project*

| No | Assumption |
|---|---|
| 1 | During the implementation stage of this project, it is assumed that the vector $\vec{B}$ rightfully represents human web browsing behaviour and is able to gain access to the web client. This means that, once the implementation is completed, the result should demonstrate that the client has passed the verification and therefore, can be allowed to enter the website. |
| 2 | During the implementation stage of the project, the testing data represents the proof y that is transmitted to the backend server-verifier by the client-prover. |
| 3 | The entries of proof $y$ have to be hashed cryptographically to preserve the privacy of the client. However, during the implementation phase of this project, the testing data (proof $y$) is not hashed. Instead, it is multiplied by a scalar $x$, which represents the privacy preservation property of the zkML. |
| 4 | The client-prover is always assumed to be dishonest and hence, is not trusted. They must prove successfully that they possess the right behavioural data that falls into the accepted range of the model to gain access to the website. |

## 3.2 Research

In order to design the ZKML, it is essential to review the literature on the four key subjects: human web browsing behaviour, Google's reCAPTCHA, Zero-Knowledge Proofs (ZKP) and Machine Learning (ML) models.

For the web browsing behaviour, research on which characteristics define a human browsing the web is conducted and there is a search to find suitable datasets portraying both human and bot data that can be used for the implementation. Eventually, a large dataset is retrieved however, it contains data relating only to the human characteristics. As datasets consisting of both human and bot behaviour cannot be found, the one class dataset is then used.

For Google's reCAPTCHA, the research focuses on understanding how the verification process works at the client-server level and what research has been done in bypassing it. For ZKPs, the project focuses on primarily two protocols: zkSNARK, and Bulletproofs, emphasising on their properties and the how they are constructed.

For ML, four algorithms are researched, namely, K-Nearest Neighbours (KNN), Recurring Neural Network (RNN), Support Vector Machines (SVM), and One Class SVM (OC-SVM). It is also necessary to find relevant references where ZKPs are integrated into ML algorithms resulting in a ZKML system, to use as examples for how these systems are constructed and to support the validity of this project. The research conducted is all in *Chapter 2 Context*.

### 3.2.1 Requirements Analysis

Based on the research, the project's objectives were established. Next, the specification requirements of how the CAPTCHA v4 needs to be and then based on these requirements a scenario was established. This scenario set goals throughout the project of specific outputs and characteristics the system should be able to produce. During the implementation step there were assumptions set as to what the model's output should be and how the new unseen data used later on to test the performance of the code, should compare with the results of the dataset used originally. A crucial aspect of the implementation was to make sure the training and testing sets were properly scaled, the inflation of the data does not impact the outcome, and most importantly, that the privacy preserving property of the ZKML is represented.

## 3.3 Design

### 3.3.1 Scenario

Drawing inspiration from how reCAPTCHA works (as described in *Chapter 2 Context 2.1*), as well as the standard ZKP protocol flow, the scenario presented in *Chapter 4 Results 4.1.1* is designed. This process began with a low-level design, having to do with the basics of how some prover can demonstrate their humanity to the verifier, which later leads into a high-level concept where the client becomes the prover, the reCAPTCHA backend server the verifier, and intricate details about how the verification is established. Two ZKP protocols, zkSNARKS, Bulletproofs proofs, are then evaluated on whether they are suitable for this scenario, based on their properties and studies that compare them. These protocols are described in detail in *Chapter 2 Context 2.4.*

### 3.3.2 ZKP Suitability

The research that was conducted and the properties of each ZKP are considered to help determine which is the most suitable ZKP protocol for this project. Specifically, the paper by El-Hajj and Oude Roelink (2024) compares zk-SNARKs with Bulletproofs by evaluating the proof size and the speed of both the proof generation and verification algorithms. Zk-SNARKs have a significantly small proof size of 288 bytes, quick proof generation of 2300ms and fast verification of 10 ms. On the other hand, Bulletproofs have a proof size of 1,300 bytes, proof generation time of 30,000ms and 10ms for verification. Additionally, the study did not find any differences in the security of the two protocols. Therefore, while the verification time is the same for both, it is evident that zk-SNARKs are much more efficient than Bulletproofs while providing the same level of security.

Ultimately, the zk-SNARK protocol was chosen as they are:

- Non-interactive due to the output of the $Gen$ algorithm.
- Efficient as their proof size is small and their proof and verification times are quick.
- Secure with a long history of applications into ML to preserve privacy.

The non-interactivity is crucial as based on the project's assumptions; the client is not trusted hence any communication with them cannot be trusted either. To add, this property allows for the frictionless ability of the reCAPTCHA v3 to continue. The efficiency is incredibly necessary as the speed of the proof and verification have to be quick to accommodate all the web clients that may enter a site. Lastly, the security is a main priority of the proposed ZKML system that ensures that the web client's behavioural data is not accessible to the reCAPTCHA backend server and therefore Google. These are all the reasons that make zk-SNARKs the ideal ZKP for the specific scenario this project focuses on.

### 3.3.3 ZKML Construction

The ZKP is constructed by first defining the statement that has to be proven. This statement is then turned to a circuit, represented by mathematical equations while at the same time diagrams are being created to depict how the prover connects with the verifier to prove this statement. Particularly, the construction is split into three parts: the setup algorithm, the proof generation algorithm and the verification algorithm, all having to do with the zk-SNARK. Once this is done, a lengthier exploration of which ML algorithms can be suitable for the verification algorithm is completed and the project follows to the implementation.

### 3.3.4 ML Suitability

The four ML algorithms considered are: RNN, KNN, SVM and the OC-SVM, described in *Chapter 2 Context*. The chosen dataset that is described in the section below, contains only one classification, meaning that all the entries represent benign data and there is no representation of malicious data. As both KNN and RNN can classify data based on two categories, only the SVM model is able to be trained on a one-class dataset. Research also shows that one-class SVM can be used for anomaly detection (Amer, Goldstein and Abdennadher, 2013) and (Tax, Duin, 2003).

While the other ML algorithms were considered, ultimately, due to the nature of the dataset, only the one class SVM proves to be suitable enough to be trained on a dataset containing a single class.

Following this, the ZKML is designed as described in the following chapter, *Chapter 4 Results*.

## 3.4 Implementation

The implementation phase transforms the theoretical concept of the verification algorithm of the ZKML, into a proof of concept. It specifically focuses on processing the data, defining and training the OC-SVM model, deriving the range of the decision function, and comparing the testing and training sets using a Euclidean distance metric. If these distances are within the model's range, then the testing data is considered similar to the training data and can be classified under the same class as the dataset. This is done using the Python programming language within a Jupyter Notebook environment. The source code is available in Appendix B and C and in the submission files of the project.

### 3.4.1 Dataset features

The data used in this project is taken from (Sanchez et al., 2020) and consists of the behavioural data of 12 different users on their personal computer devices over the course of 55 days. This data is proven to be suitable for this project as it contains a large amount of entries, but with only one class which relates to the behavioural characteristics of each user. It is also deemed suitable as its' features represent the activity of real humans and this is able to meet Objective 1.

The following tables are created with information directly from the README.md of the downloaded folder. They describe the keys and features of the dataset that are being studied. (Sanchez et al., 2020)

*Table 3: The keys the study was monitoring (Sanchez et al., 2020)*

| Name | Keys |
|---|---|
| list_all_keys | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g, h, i,j, k, l, m, n, ñ, o, p, q, r, s, t, u, v, w, x, y, z, ´,`, ', ", ç, \^, º, @, \$, \%, \&, /, (, ), =, +, \|, 'leftwindows', 'crtl', 'shift', 'capslock', 'tab', º, ª, \textbackslash, \#, 'esc', 'f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12', 'PrtSc', 'insert', 'delete', 'home', 'ind', 'pageup', 'pagedown', 'numlock', \}, \{, -, _, '.', ',', [, ], *, <, >, 'space', 'tab', 'inter', 'rightctrl', 'rightshift', 'backspace', 'atlgr', 'alt', 'left', 'right', 'up', 'down', 'rightarrow', 'leftarrow', 'uparrow', 'downarrow' |

| | |
|---|---|
| list_chars | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g, h, i, j, k, l, m, n, ñ, o, p, q, r, s, t, u, v, w, x, y, z, ç |
| list_space | -, _, '.', ',', /, \&, +, <, space, tab, enter, (, ), =, \|, \textbackslash, \# |

*Table 4: The generated features of the dataset (Sanchez et al., 2020)*

| No | Name of feature | Description |
|---|---|---|
| 1 | timestamp | The timestamp when the entry was generated. |
| 2 | keystroke_counter | Total number of keystrokes by the user during the window. |
| 3 | erase_keys_counter | The amount that the erasing keys were pressed. Includes 'delete' and 'backspace' keys. |
| 4 | erase_keys_percentage | The proportion of erasing keystrokes used compared to the overall number of keystrokes. |
| 5 | press_press_average_interval | Average time between two consecutive keystrokes in milliseconds |
| 6 | Press_press_stddev_interval | Standard deviation of time between consecutive keystrokes. |
| 7 | Press_release_average_interval | Average time from pressing to releasing a key during the time window. |
| 8 | Press_release_sttdev_interval | Standard deviation of time between pressing and releasing a key |
| 9 | Word_counter | Total number of words written by the user during the time window |
| 10 | Word_average_length | The average length of words typed during time window |
| 11 | Word_stddev_length | Standard deviation of word length typed during time window |
| 12 | Word_length_N | histogram of word lengths (from 1 to 11+ characters) |
| 13 | Keystrokes_key_K | Count of keystrokes for each key |
| 14 | Press_release_average_K | Average time between pressing and releasing each key |

| | | |
|---|---|---|
| 15 | Digraph_counter_KK | Count of two-key combinations (digraphs) typed by the user |
| 16 | Digraph_average_time_KK | Average time between pressing two consecutive keys (digraphs) |
| 17 | Click_speed_average_N | Average time for each mouse button click (left, right, double, or middle) in milliseconds. |
| 18 | Click_speed_stddev_N | Standard deviation of mouse click time. |
| 19 | Mouse_action_counter_N | Count of mouse actions (e.g., clicks, scrolls, movements) |
| 20 | Mouse_position_histogram_N | Count of mouse events in each of 9 screen quadrants. |
| 21 | Mouse_movement_direction_histogram_N | Number of mouse movements in each direction (8 cardinal points) |
| 22 | Mouse_movement_length_histogram_N | Histogram of mouse movement lengths based on screen size. |
| 23 | Mouse_average_movement_duration | Average duration of mouse movements. |
| 24 | Mouse_average_movement_speed | Standard deviation of mouse movement speed. |
| 25 | Mouse_average_movement_speed_direction_N | Histogram of average mouse movement speed per direction. |
| 26 | Active_apps_average | Average number of applications active during the time window. |
| 27 | Current_app | Application in the foreground when the vector was created. |
| 28 | Penultimate_app | The penultimate application in the foreground during the time window. |
| 29 | Changes_between_apps | Number of switches between foreground applications. |
| 30 | Current_app_foreground_time | Duration the current app has been in the foreground. |
| 31 | Current_app_average_processes | Average number of processes the foreground app had active. |
| 32 | Current_app_stddev_processes | Standard deviation of processes in the current app. |

|  |  |  |
|---|---|---|
| 33 | Current_app_average_cpu | Average CPU percentage used by the foreground app. |
| 34 | Current_app_stddev_cpu | Standard deviation of CPU usage by the foreground app. |
| 35 | System_average_cpu | Average CPU percentage used system-wide. |
| 36 | System_stddev_cpu | Standard deviation of total system CPU usage. |
| 37 | Current_app_average_mem | Average memory percentage used by the current app. |
| 38 | Current_app_stddev_mem | Standard deviation of memory used by the current app. |
| 39 | System_average_mem | Average system memory usage percentage. |
| 40 | System_stddev_mem | Standard deviation of total system memory usage. |
| 41 | Received_bytes | Bytes received through network interfaces during the time window. |
| 42 | sent_bytes | Bytes sent through network interfaces. |
| 43 | USER | Label identifying the user, represented as a number from 0 to 11. |

### 3.4.2 Data preprocessing

Each user's data is stored in an individual CSV file, with user 4 consisting of 10114 rows and 12048 columns. Realistically, each data from the 11 users' needs to be merged together and then ran through the code. However, this will result in a merged dataset of 5.05GB and running this becomes computationally expensive and beyond the computational storage of the author's personal device. Therefore, for the full implementation and development of the source code, only one user's (user 4) dataset is used. Users 0, 2, 3, 4, 5, 6, 8, 9 and 10 are implemented on a part of the to justify the range chosen for user 4. Users 9 and 10 are also implemented on the full source code for testing purposes later on.

Most of the data is numeric, except for two non-numeric columns, namely, no.27 and no.28 in Table 3. The entries of the last feature, no.43 in Table 3, are marked with the number of the user that the csv file belongs to. For example, for User 5 all the entries in that column are of the number 5. These 3 non-numeric columns do not contribute to the computations but complicate them therefore they were removed. Entry no.1 is the timestamp of the entries, a column that can be used as index but to simplify things for this project, is removed.

## 3.4.3 How key results are achieved

The dataset used to achieve the key results is of the user 4, from the 11 users from the study. The dataset is pre-processed as described in 3.4.2, and the training and testing sets of the model are defined. Both sets are the dataset itself, denoted as B, with the testing set being also multiplied by a scalar $x$. This represents the privacy preserving aspect of the ZKML, and while it is not directly as how it would be in a thorough implementation within a CAPTCHA environment, it is only symbolic in this case.

Both sets are scaled in a MinMaxScaler instance (scikit-learn, 2024), setting the values between 0 and 1. The training set is the fitted into the OC-SVM and its hyperparameters tuned so that the kernel is the $rbf$, the $\gamma - "gamma"$ set to the default, $scale$ and the $\nu - "nu"$ parameter as “0.1”.

Afterwards, the decision function is derived for the scaled training data and the minimum and maximum values are calculated. The reason for this is to define the acceptable range within which the Euclidean distances between both sets must fall. These values are then visualised in a scatter plot, with the negative values identified as outliers. Due to the distribution of values being spread out with no areas of high density, the plot is then compared with the decision function plots of 8 other users in the study. This comparison justifies defining the model’s range between the first positive minimum value and the maximum value.

Subsequently, this range is multiplied by the scalar x to counterbalance the multiplication of the testing with the same scalar. If this is not done, the Euclidean distances will not be able to fit into the range as they will become “inflated” from the multiplication of the testing set.

Then, the Euclidean distances between the training and testing sets are calculated, only after both sets are transposed to make the features become rows. This is due to the code computing the distances by rows when in this specific case the author wants the features to be compared. These distances are tested to see if they fall within the predetermined range. The outputs are in a Boolean array with “True”/”False” statements. These statements are counted, and the majority classifies the testing data as either “1” – “True” or as “0” – “False”. This 0 and 1 classification is also the output that is expected from the verification algorithm of the ZKML.

As the dataset contains human browsing characteristics then a classification of “1” can label the web client as a human. Despite this, as the dataset only contains one class and no data referring to activity of bots and the model has not seen any bot data then it cannot definitively state that an output of “0” means that the web client is a bot. In the project, a classification of “0” is described as “potentially non-human”. This proves to be a limitation.

Notes:

1. Throughout the results section, none of the decimals are rounded to ensure the outmost accuracy.
2. Users 1, 7 are not examined in this project, as they are the largest with 1.42GB and 1.87GB respectively, and proved to be computationally complex for the capabilities of the author's personal device.
3. When comparing the plots of 8 other users, the same lines of code are used on all of them.

## 3.5 Testing and Evaluation

### 3.5.1 Testing

The testing aspect of the project comprises of testing the legitimacy of the code and model. This includes running it on unseen users of similar size namely, Users 9 and 10. Their final outputs are then compared to the results of user 4 and conclusions are made regarding the validity of the model and whether the desired outputs were produced. These datasets are ran on the exact same lines of code as the user 4.

### 3.5.2 Evaluation

The evaluation of the project consists of an exploration as to whether the use of a ZKML into an reCAPTCHA system can be beneficial. This is done by first evaluating its' design and implementation overall. The framework's advantages and disadvantages are then investigated, as well as its' position within relevant literature. Lastly, the potential security is analysed by presenting a threat model that outlines any potential attacks, the type of adversaries, ways to mitigate the attacks and their likelihood and severity.

# 4 Results

## 4.1 Overview

### 4.1.1 Proposed scenario

This project's proposed high-level scenario is displayed in Figure 7 and described below. The web client will be referred to as client-prover and the reCAPTCHA backend server as server-verifier.

A reCAPTCHA verification system is applied on a website and a web client wants to enter the site. The client wants to prove to the server that they are human and therefore, enter the website.

The main components of this framework are:

- The web client – the prover in the ZKP context
- The website server,
- The reCAPTCHA backend server – so the verifier in the ZKP context
- The trusted authority

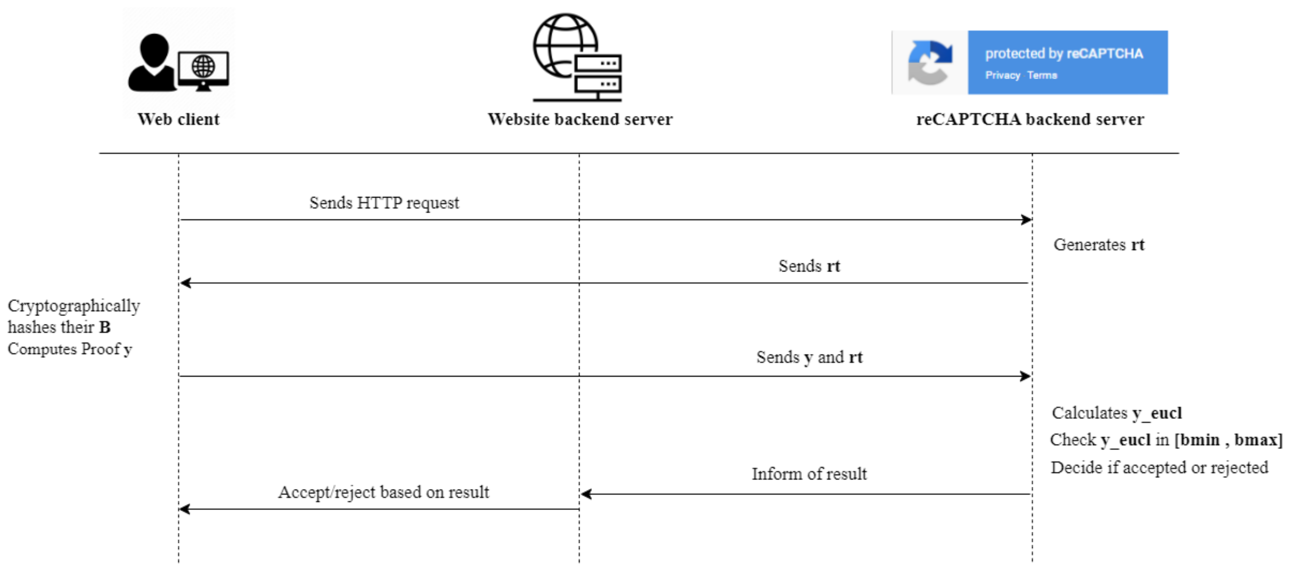


*Figure 7: Protocol flow of proposed scenario*

a. The client-prover initiates the process by sending an HTTP request to the server-verifier, and in response the server-verifier sends back a response token ($\boldsymbol{rt}$).

b. Subsequently, the client-prover applies a cryptographic hash function on their behavioural data $\vec{\boldsymbol{B}}$, represented as $\boldsymbol{H}$ and in combination with the proving key $\boldsymbol{pk}$ in the proving algorithm of the zkSNARK ($\boldsymbol{Proof}$), constructs the proof $\boldsymbol{y}$.

c. This computed $\boldsymbol{y}$ is transmitted to the server-verifier alongside the $\boldsymbol{rt}$ that was previously received.

d. The backend-verifier then examines whether the Euclidean distances between the features of $\boldsymbol{y}$ with the features of the model's training set are within the set range of the model, denoted as

($\boldsymbol{bmin}, \boldsymbol{bmax}$). Afterwards, if it is determined that the majority of these distances are positioned within this range, the outcome results in the client-prover becoming eligible to access the website. Conversely, if the majority of these distances are not within the set range, then the client is denied access.

e. This result is then sent to the website server and based on this result, the website either a accepts or rejects the web client-prover, concluding the entire verification process.

## 4.2 Design

This section presents the results derived from the design stage of the project.

The research conducted in *Chapter 2 Context 2.3.2* outlines that the zkSNARK protocol contains three algorithms: $Gen$, $Proof$ and $Verif$. In this project, the design stage defines all three algorithms theoretically while the implementation demonstrated a proof of concept for the $Verif$ algorithm.

The statement $m$ being proven is that the client wants to demonstrate that their behavioural data defines human web browsing activity without explicitly stating the contents of that data.

More specifically, for a language $L$ that represents all the inputs that satisfy the success of the statement, the honest prover (Client) wants to convince the verifier (Server) that some vector $\vec{B}$, is in $L$. By doing so, the client proves that they possess behavioural characteristics attributed to humans on the web.

The following definition is constructed in reference to Stephens-Davidowitz, (2023).

**Definition:** A protocol $(Client, Server)$ is a zero-knowledge succinct non-interactive argument of Knowledge (zkSNARK) protocol for the language $L$ if the protocol satisfies the following properties:

- Completeness: For every valid input $m \in L,\ Pr\ [(Client, Server)(m) = 1] = 1$

Meaning that the probability that B is in the language, so the statement has been proven successfully, is 1. In other words, if the prover is honest and has not manipulated their behavioural data $B$, the verifier will accept the proof.

- Soundness: For an input $m \notin L$, from a dishonest prover Client$^*$, then $\Pr[(Client^*, Server)(m) = 1] \leq \frac{1}{2}$

This means that if the client is an illegitimate bot and their behavioural data is not of a human, then the probability of them successfully proving the statement, is negligible.

- Zero Knowledge: For a simulator S, such that for any adversary A,

$$\Pr\left[A\left(m, S(m)\right) = 1\right] - \Pr\left[A\left(m, view^{server}(m)\right) = 1\right] \leq \epsilon(|m|)$$

The simulator S is a hypothetical algorithm, taking only public inputs to generate a "simulated transcript" of the interactions, without knowing the secret (Green, 2017) and the adversary A is a malicious verifier. $view^{server}(m)$ represents the view of the server-verifier when interacting with a prover. Overall, this property states that there exists a simulator S that can essentially generate a "fake" view of the prover-verifier interaction so that the probability that an adversary A can distinguish between a real interaction and this fake one is negligible. This means that the verifier cannot learn anything from the prover but the publicly available information.

The following diagram represents in detail how the client-server protocol flow happens.

| Prover | | Verifier |
|---|---|---|
| INPUT: $\boldsymbol{H}, \vec{\boldsymbol{B}}$ | TA | INPUT: $\boldsymbol{H}$ |
| | $Gen(\lambda, C) \rightarrow (pk, vk)$ | |
| Create hash for secret $B$<br>$H = hash(B)$ | | |
| Construct proof<br>$Proof(pk, H, B)) \rightarrow y$ | | |
| | $\xrightarrow{\quad y \quad}$ | |
| | | Calculate $sqrt(sum(x_i - y_i)^2) \rightarrow y_{eucl}$<br>Check $bmin < y_{eucl} < bmax$<br>Check $H = hash(B)$<br>$\Rightarrow Verif(vk, H, y) \rightarrow 1$ |

The circuit $C$ of the system is the following algorithm 1.

```
Algorithm 1: Arithmetic Circuit C
Private input: B
Public input: H
Output: decision bit
a. Create H = hash(B)
b. Calculate $Proof(pk, H, B) \rightarrow y$
c. Calculate $sqrt(sum(x_i - y_i)^2) \rightarrow y_{eucl}$,   $for\ i = 1, 2, 3, \dots, n$
d. Check if $bmin < y_{eucl} < bmax$
e. Verify $H = hash(B)$
f. If d. and e. hold then $Verif(vk, H, y) \rightarrow 1$; $otherwise\ reject$
```

Note: In a comprehensive implementation of the entire proposed zkML within reCAPTCHA, which is beyond the scope of this project, an output of “1” would denote human web browsing activity, whereas an output of “0” would indicate bot activity. However, the model examined in this project is trained exclusively on a dataset consisting of only one class that represents human activity. Since, no data that denotes bot activity is present, then the outcome cannot conclusively state that an output of “0” denotes bot activity.

## 4.3 Implementation

### 4.3.1 Overview

As mentioned in the section above and in *Chapter 3 Methods* the proof of concept contains the $Verif$ algorithm and the model that is trained is a one-class SVM. This coincides with assumption number 2.

The code of the implementation can be viewed in Appendix B and contains:

- The preparation of the dataset of User 4.
- The training of the model.
- The visualisation of the model's decision function.
- The definition of the boundaries of the decision function, referred to as the "range" in the previous section.
- The comparison of testing and training data using the Euclidean distance.
- The classification of the testing data by evaluating whether the Euclidean distances fall within the specified range, with the majority outcome classifying the testing data.

### 4.3.2 Key results

The key results of the implementation are presented below,

<u>Deriving the range of the model:</u>

The boundary for the model on the training set is derived by the minimum and maximum of the decision function. This range is:

- Minimum: $-20.15023390219465$
- Maximum: $52.70627165648634$

This range is quite large, so the decision function values are plotted in a scatter plot. This reveals a wider perspective of these values and their distribution, displayed in Figures 8 and 9.

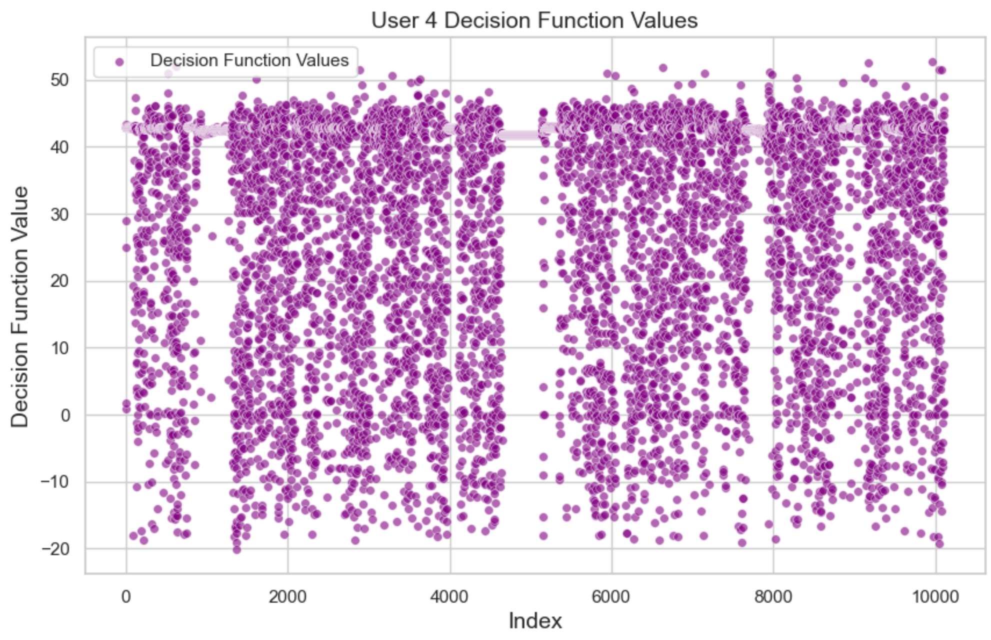


*Figure 8: Decision Function Values*

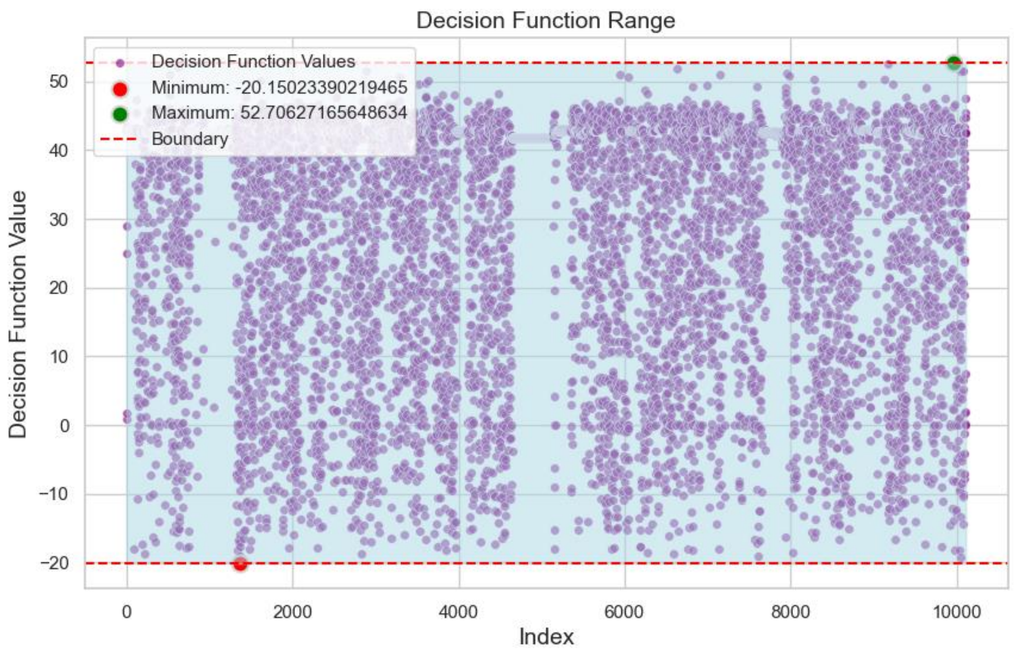


*Figure 9: Minimum and Maximum range*

From these it is clear that there are significantly less values that are negative in comparison to the positives. These negative values are considered outliers and are not considered within the range of this model. Figure 10 portrays only the positive values of the decision function.

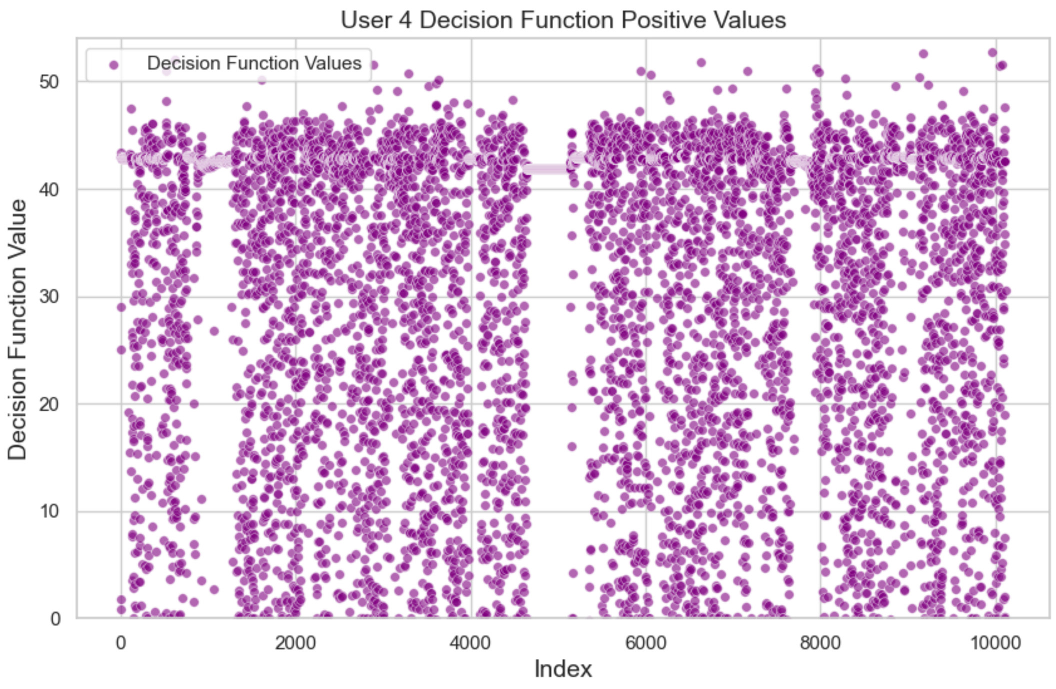
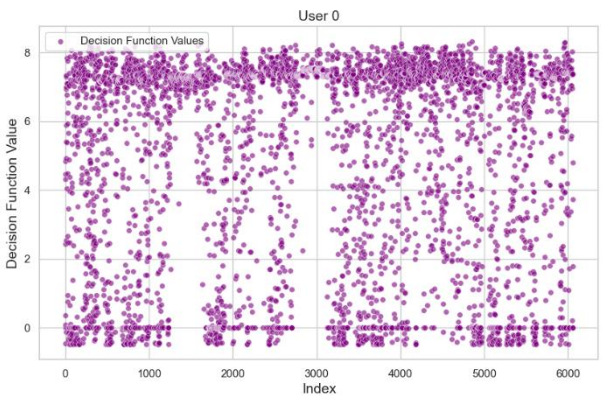
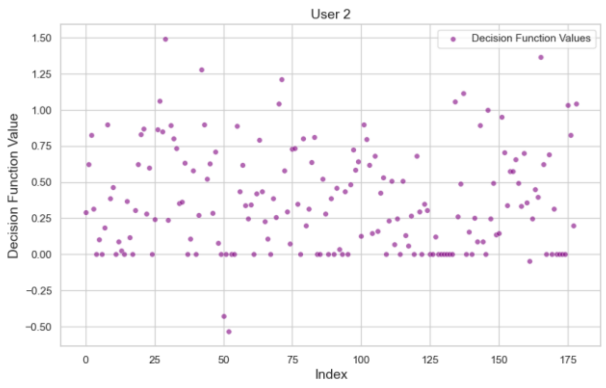
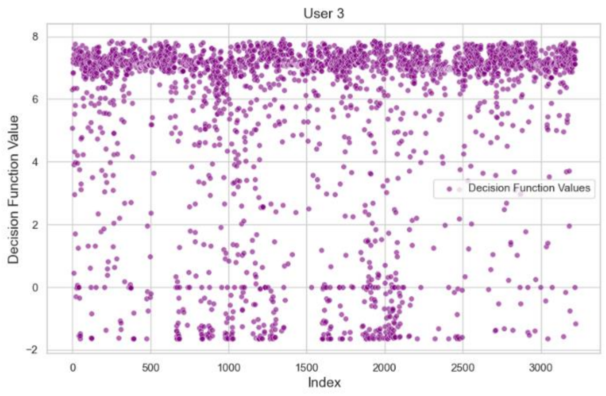


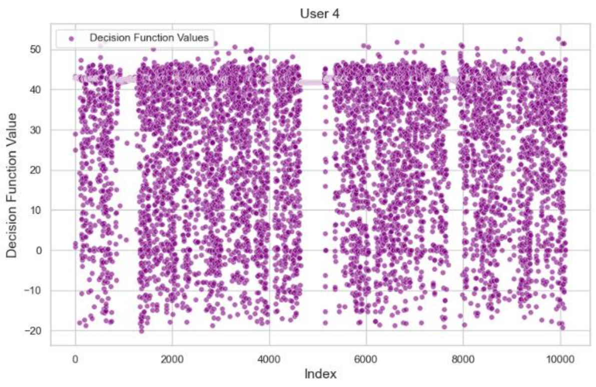
*Figure 10: Positive values*

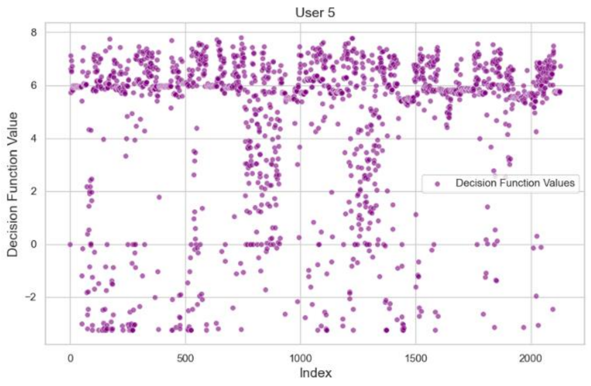
However, besides the negative values, the rest of the outliers are still clearly visible. While the area between y=30 and y=50 is denser than the rest of the plot, assuming that the outliers are between y=0 and y=30 is will not work as that area is heavily populated. To gain more perspective on this, the same line of code is ran for the other datasets within the study, namely Users 0, 2, 3, 5, 6, 8, 9 and 10 and their outputs are analysed and compared to the plot of User 4. This is seen in Figure 11.

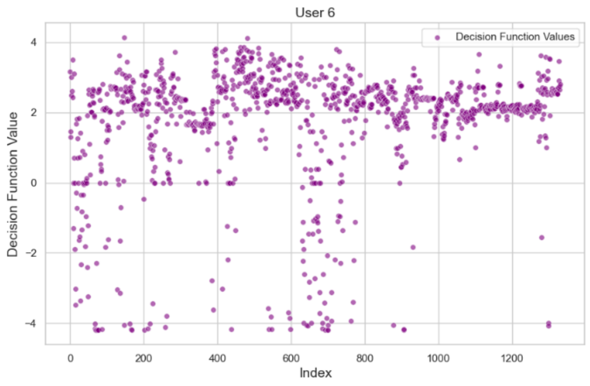

User 0
Decision Function Values
Decision Function Value
Index

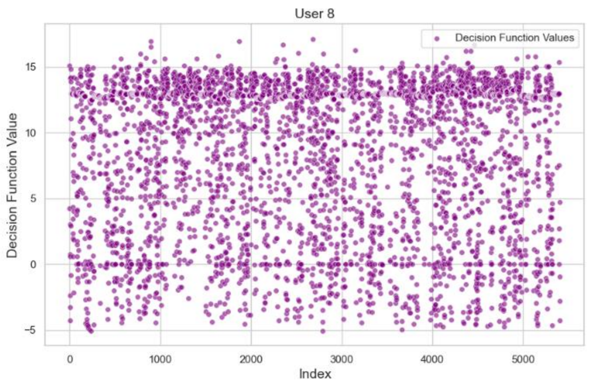

User 2
Decision Function Values
Decision Function Value
Index

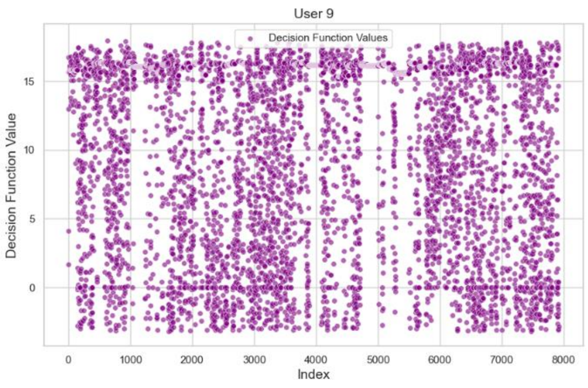

User 3
Decision Function Values
Decision Function Value
Index


User 4
Decision Function Values
Decision Function Value
Index


User 5
Decision Function Values
Decision Function Value
Index


User 6
Decision Function Values
Decision Function Value
Index


User 8
Decision Function Values
Decision Function Value
Index


User 9
Decision Function Values
Decision Function Value
Index

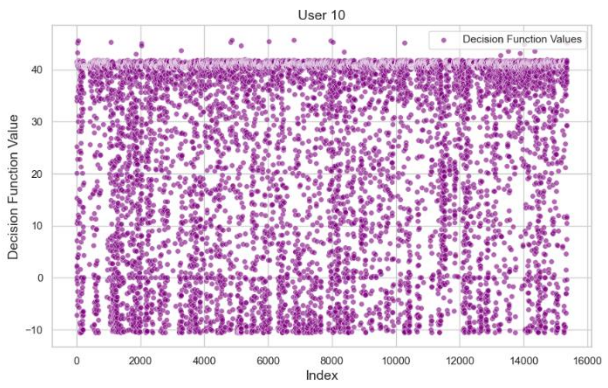


*Figure 11: Other user's decision function outputs*

Table 5 presents the concentration of data in each plot for each user, arranged in ascending order along with the respective sizes of the datasets. Including the dataset sizes provides context and is used as a justification as to which users are given more consideration. This approach assumes that users with larger datasets will yield more accurate results in showing where the boundary lies, as they have more data.

*Table 5: Range where data is concentrated.*

| User | Range (on y axis) | Size of dataset |
|---|---|---|
| 10 | 0-40 | 531MB |
| 4 | 0-50 | 350MB |
| 9 | 0-15 | 274MB |
| 0 | 6-8 | 209MB |
| 8 | 10-15 | 187MB |
| 3 | 6-8 | 111MB |
| 5 | 5-8 | 73.8MB |
| 6 | 1-4 | 46.3MB |
| 2 | Not enough data | 6.53MB |

There is a clear relationship between the sizes of the datasets and the ranges displayed. Larger datasets, such as those of users 10, and 9, show wider range of values, meanwhile smaller datasets like those of users 5, 6, and 2, exhibit a much more limited range. The close similarity of the range of users 9 and 10 justify the decision to use a broad range for user 4 as well, displayed in Figure 10.

As the minimum value cannot be an outlier, then the range of the model for user 4 spans between the first positive value and the maximum value. This new minimum value is the first positive point after the negative minimum value.

- New minimum value: $9.203778539301766e-07 \approx 0.0000009203778539301766$

This adjustment is reflected in Figure 12, alongside the negative minimum.

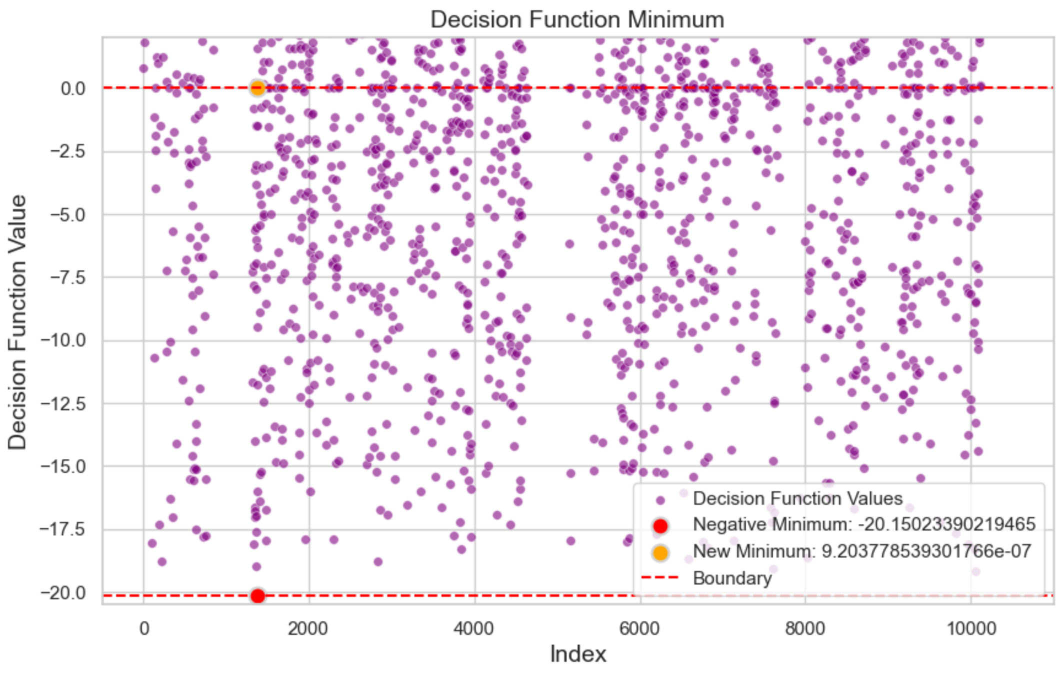


*Figure 12: New minimum value*

Hence, the new range is:

$$[\,0.0000009203778539301766\ ,52.70627165648634\,]$$

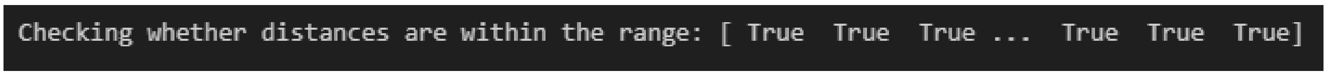
As discussed in *Chapter 3 Methods 3.4.3*, since the testing data is multiplied by a scalar $x$ (where $x = 50$) to represent the privacy preserving property of the zkSNARK, the actual range of the model is also multiplied by this scalar to counterbalance the effect of the multiplication. Otherwise, the values would be inflated, and the later on the Euclidean distance would not fit within the range regardless of whether $\vec{B}$ is suitable or not. Therefore, the product of this becomes the new range,

$$[\mathbf{0.00046018892696508 83}\,,\mathbf{26353.13582824317}]$$

<u>Comparing the testing and training sets:</u>

The testing data is compared to the training data, using the Euclidean distance metric, producing 145,130,209 distances.

These distances are then evaluated against the previously defined range to determine whether they fall within it. The results are presented in a Boolean array, with only a few displayed in the output, as shown in Figure 13.



```
Checking whether distances are within the range: [ True  True  True ...  True  True  True]
```

*Figure 13: Output of evaluation*

These “True” and “False” statements are then counted, with the majority determining the classification of the testing data. This classification is in the form of a decision bit – “1” and “0”. These results are displayed in Table 6 below.

*Table 6: Classification Results*

| Classification | Count | Percentage |
|---|---|---|
| 1 | 115318609 | 79.46% |
| 0 | 29811600 | 20.54% |

Thus, the model concludes that “1” constitutes the majority at 79.46% with a count of 115,318,609, while “0” accounts for 20.54% with a count of 29,811,600. This classifies the testing data as “1”. As mentioned in the section *4.2*, an output of “1” implies a high likelihood of human web browsing activity.

As per the 2nd assumption outlined in section *3.1.2,* the testing data is also the proof $y$ for the proving phase of the zkSNARK. As a result, this proof $y$ is then classified as having human characteristics. This means that the first assumption in section *3.1.2* is satisfied as expected, confirming that the $\vec{B}$ used in the implementation is verified to contain data that defines human web browsing behaviour.

## 4.4 Testing & Evaluation

### 4.4.1 Testing

The data of users 9 and 10 are put through the system and their results are compared to the results in the previous section, *4.3.2 Key Results.* The source code is in Appendix C.

Note: All the methods remained the same as for the User 4 file with the exact same code. The purpose of this testing is to make sure the model produces the same output for different, unseen data.

Deriving the range:

The values derived from the decision function for users 9 and 10 are visualised in Figure 11 in the previous section, and Figure 14 below, illustrates the positive values only.

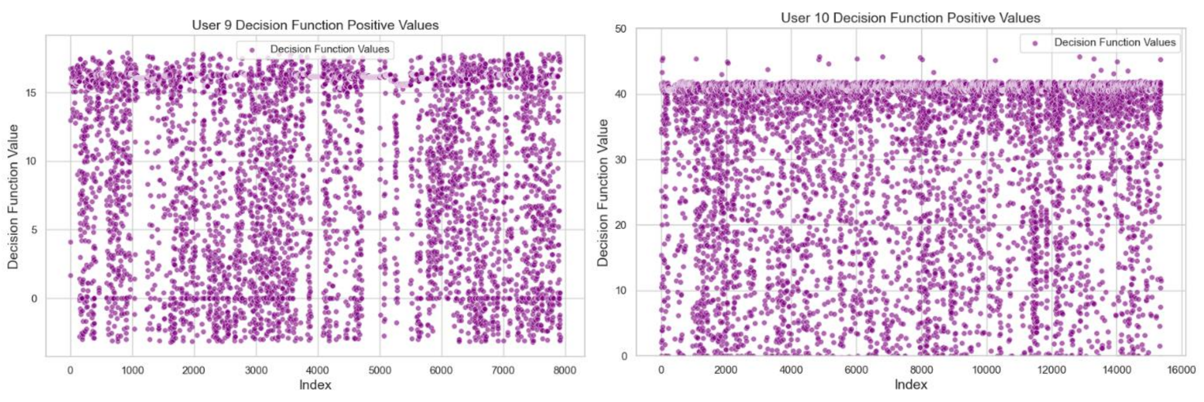


*Figure 14: Users 9 and 10 Positive Values*

The range is defined in the exact same manner as with User 4, to facilitate a fairer comparison between the three. Consequently, the ranges for the decision function that exclude the negative minimum and take into consideration the multiplication by $x$ as shown in Table 7. The ranges that will be used is highlighted in blue.

*Table 7: Ranges*

| User | Minimum (after x multiplication) | Maximum (after x multiplication) |
|---|---|---|
| User 9 | $\mathbf{0.00019827711756903454}$ | $\mathbf{903.6179453863418}$ |
| User 10 | $\mathbf{0.00043857996354645934}$ | $\mathbf{2283.5061037594132}$ |

<u>Comparison metric:</u>

Table 8 presents the count of "1" and "0" outputs obtained by examining whether the Euclidean distances between the testing and training sets fall within the range specified in Table 7.

*Table 8: Classification Results*

| **User** | **Classification** | **Count** | **Percentage** |
|---|---|---|---|
| 9 | 1 | 116691086 | 80.4% |
|  | 0 | 28439123 | 19.6% |
| 10 | 1 | 97785474 | 67.38% |
|  | 0 | 47344735 | 32.62% |

User 9 has a count of 116691086 distances with an 80.4% for classification "1" and a count of 28439123 distances consisting of 19.6% for "0". User 10 shows a count of 97,785,474 distances representing the 67.38% for "1" and the classification of "0" with a count of 47,344,735 representing a 32.62%.

While both users classify the testing data with a "1", user 9 shows higher tendency towards while user 10 shows a lesser tendency to "1" but still remains the majority.

<u>Comparison with the results of user 4:</u>

*Table 9: Comparison of results*

| **Classification** | **User 4** | **User 9** | **User 10** |
|---|---|---|---|
| 1 | 79.46% | 80.4% | 67.38% |
| 0 | 20.54% | 19.6% | 32.62% |

From Table 9, it is evident that the classification of "1" is significantly higher for user 9 with 80.4%, with user 4 closely following at 79,46%.  In contrast, user 10 shows a lower tendency towards the classification, with only 67.38%. The difference between the users 4 and 9 is very close however user

10 varies slightly from both. This is most likely due to the boundary range having to be adjusted for user 10 due to it being a larger dataset and containing more data.

The similarity between the findings of all three users, justifies the legitimacy of the model's code.

### 4.4.2 Evaluation

The zkSNARK protocol has been applied solely in a theoretical context, meaning that a comprehensive security evaluation of whether the proposed protocol is complete, sound and zero-knowledge cannot be thoroughly examined. Nevertheless, other theoretical aspects of the protocol can be explored. This includes positioning the protocol within relevant research, outlining its advantages and disadvantages while identifying areas for improvement and presenting a potential threat model.

- Project's position in literature:

As discussed in *Chapter 2 Context 2.4*, there is an abundance of ZKML applications all for different purposes. Yampolskiy (2012), proposes an idea that is opposite to what this project is trying to achieve. The paper combines ZKPs with CAPTCHA, referenced as "SuperCAPTCHA", that can only be solved by artificial intelligence (AI). This method demonstrates that if the SuperCAPTCHA is solved then this proved the existence of the AI without disclosing any specific details about how it operates. While this approach achieves the opposite of the purpose of the ZKML of this project, it remains relevant literature that merges ZKP and CAPTCHAs, as early as 2012.

There has been another relevant study regarding an android service, ZKSENSE, which is a verification system based on a ZKP that investigates whether users are human. This is possible due to the motion sensor data of the device, and the overall aim is to protect privacy. The paper claims a 92% accuracy in a significantly short time, 3 seconds, quicker than CAPTCHAs (Querejeta-Azurmendi et al., 2021).

While there is literature exploring the mix of ZKPs and CAPTCHAs, and new verification systems based on ZKPs, there is no existing literature on implementing ZKP to improve a CAPTCHA verification system to prove that a user is human without disclosing specifics about how the inner working of the system. This absence highlights a gap in existing research and places this project in a unique position.

- Regarding the advantages and drawbacks on the zkML:

Positives:

1. The zkSNARK's $Gen$ algorithm outputs, turn the protocol into a non-interactive protocol. This is crucial as the client is dishonest and is not trusted so communication with them cannot be guaranteed to be trusted.
2. In addition, the cryptographic hash function adds the privacy protective property to the protocol, ensuring that by the time the proof y is sent to the verifier, they cannot retrieve the

original B. As there have been many concerns over the years regarding the privacy of what happens with the data that google collects, this method will be able to mitigate these issues.

3. No score is returned to the client as it currently is with reCAPTCHA v3. Getting the score sent back to the client can help them adjust different parameters in their behavioural data, to try as best as possible to mimic a human on the web. Similar research was done by Tsingenopoulos et al., (2022) and Akrout et al., (2020), described in *Chapter 2 Context*, by using reinforcement learning to bypass v3 with a high success rate. This opens up the potential for a model inversion attack (expanded in detail in Table 11). This issue can be eliminated by just providing the client and web host with the classification without giving an approximation as to how close to getting classified as human the client is, meaning for the attack to succeed the parameters would have to be adjusted blindly, making the system more complicated and raising the level of difficulty to bypass it.

Drawbacks:

1. The need for a trusted setup opens up the system to vulnerabilities. This trusted setup needs to be setup securely by a secure third-party leading discussions for future work.
2. Another consideration is that the client-prover has to compute a lot of things. They have to calculate the hash of their behavioural data and construct their proof. In early iterations of this project ways to avoid this were considered, however they do not fit in the standard prover-verifier exchange and flow of a ZKP, which in turn would complicate how the verification was going to happen. A proposal for this, would be to build an algorithm within the website's server that will calculate all this on its' own. However, this brings on another issue of having to conceal this algorithm from the hosts. All in all, this calls for further examination in future work.

- Implementation:

The implementation is compared using three metrics, runtime, the average CPU usage and memory usage from the first line of code until the last. The code is also evaluated to see how it performs with the findings of the testing. For context, this is based on the implementation of User 4's data which is 350MB. The result are seen in Table 10.

*Table 10: Metrics and their results*

| Metric | Result |
|---|---|
| Runtime | ≈ 31.13 minutes |
| CPU Usage | 4.6% |
| Memory Usage | ≈ 3.407GB |

The runtime of approximately 31.13 minutes is long with the lines of code taking the longest runtime are where the large training set (350MB) was fitted to the OC-SVM model, and for calculating the Euclidean distances between the training and testing sets.

A CPU usage of 4.6% indicates that the CPU was not heavily utilised. This means that the code is not computationally expensive on the CPU but most likely on the memory. This indicates the need for future work to focus on the improving and utilising the CPU a lot more. A proposal is to adapt a parallelisation processing technique and optimising the operations like the Euclidean distance computations and model fitting.

The peak memory (RAM) usage of the device during the execution was 3.407GB which is quite large due to the large dataset the model is working with. The potential lines consuming memory are the loading of the dataset, the model fitting, and calculating the distances.

The code and model seemed to work well as the assumption 1 that the dataset B is indicating human activity is satisfied. The findings in the testing section proved that the range that was derived for user 4 was valid as the basis of this decision was on the fact that users 9 and 10 had similar ranges. Running these two datasets through the code proved that the testing was also deemed of human as per the assumption 1. It would have been more efficient to merge all 11 datasets together however, as this came up to 5.05GB, it proved to be computationally expensive based on the author's device's capabilities. It would have also been helpful to analyse and run the code on the largest datasets such as User 1 and 7 but as these were more than 1GB each, which proved to be a limitation.

Overall, the code was working and achieved the desired outputs and assumptions of the implementation of this project. Testing proved that this code worked for other datasets of similar or larger sizes. However, there were many ways that the code could have been optimised by using different techniques like parallelisation to make a better usage of the CPU as to not make it so memory intensive, potentially reducing the runtime too.

- Security considerations

The following describes a potential threat model with attacks regarding the proposed ZKML.

*Table 11: Threat model of the ZKML (0 is low, 10 is high)*

| **Attacks** | **Threat Actors** | **Mitigations** | **Likelihood** | **Severity** |
|---|---|---|---|---|
| Model Inversion Attack<br><br>The adversary tries to learn information about how the model works by changing some of their behavioural data until they are able to pass the verification and gain access to the website. | Anyone with malicious intent/bots | 1.Ensure the output of the verification algorithm is not sent back to the host as it is currently in reCAPTCHA v3.<br>2.Regular monitoring for any vulnerabilities and their patching to ensure nothing slips through the cracks. | 4 | 10 |
| Replay attack<br>-<br>Adversary intercepts the computations of the prover and sends them to the verifier as their own. | Adversaries with malicious intent | 1. There needs to be a thorough examination as to how these computations will be calculated from the web client's perspective to make sure that neither the client nor the web host can be aware of what is actually being computed.<br>2. The trusted setup needs to be secure, conducted by a trusted party, to ensure that it cannot be exploited. | 7 | 10 |
| False proof generation<br>-<br>Adversaries may get verified by using fake proof y. | Adversaries with malicious intent, we clients with technical knowledge | Meticulous planning to ensure the zkSNARK is fully sound, meaning a dishonest prover will not be able to convince the verifier that the statement is true when in reality it is not. | 6 | 10 |
| Lack of zero-knowledge<br><br>The ZKML system may leak more information that it is supposed to, failing one of the ZKP's properties, giving information about the model and how the verification happens. | Developers of the ZKML that did not conduct a proper security evaluation of the system. | Need to ensure that the protocol satisfies one of its main properties of being zero-knowledge in its implementation. | 6 | 9 |

## 4.5 Human Behaviour characteristics

As mentioned previously reCAPTCHA v3 takes in consideration the web client's behavioural data to filter the legitimate from illegitimate users. More specifically, it monitors the "user's static information, including source IP addresses, user-agent header and browser cookies and their dynamic information like how the users interact with the target website, such as keyboard typing and mouse movements” (Kotaro, 2022).

The dataset that this project is using contains 39 overall features. A description of these is in *Chapter 3 Methods 3.4.1.* Categorically, they contain keystroke metrics such as user typing behaviour and number of keystrokes as well as mouse behaviour including the click speed and movement direction. The dataset contains features relating to CPU/memory consumption, the number of active processes but also it monitors network traffic by the number of bytes sent and received.

Therefore, the human characteristics of the project, are the features of the dataset as they are corroborated by the research and what data Google tracks in their reCAPTCHA version 3.

# 5 Discussion

## 5.1 Results in comparison to objectives

| | |
|---|---|
| **Objective 1:** Identify what characteristics determine human web browsing behaviour. | This was achieved by the research conducted in *Chapter 2 Context 2.2* and corroborated by the chosen dataset. |
| **Objective 2:** Identify which parameters of non-interactive ZKPs can be implemented into CAPTCHA. | The zk-SNARK is found to be the most suitable due to its fast proof and verification times, and the small proof size. |
| **Objective 3:** Identify which characteristics of ML algorithms can be implemented into CAPTCHA. | Due to the dataset only containing one class, the most suitable ML algorithm is the One-Class SVM. |
| **Objective 4:** Evaluate how effective the integration of ZKPs and ML models (ZKML) is when implemented into CAPTCHA. | The ZKML framework is analysed theoretically, a potential threat model is conducted, and the code of the implementation is evaluated. |

### 5.1.1 Objective 1

This objective is to define what constitutes human web browsing behaviour. This was achieved by researching what reCAPTCHA v3 tracks and if the chosen dataset's features are corroborating this as this data is collected from real humans browsing the web on their personal devices. Therefore the human web browsing characteristics are the features of the dataset and the objective is satisfied.

### 5.1.2 Objective 2

This objective is examining which ZKP is most suitable for the scenario of this project. This objective was met using extensive literature on how the two proposed ZKPs operate, their properties and they were all thoroughly examined to see how they can be adapted to the scenario.

The zkSNARK proved to be the most suitable as it is a non-interactive proof and it is the fastest with the smallest proof. A drawback is that it requires a trusted setup, however with thorough planning this can be handled in a secure way. Another issue are the logistics of how the prover computes all the computations with a suggestion in *Chapter 4 Results 4.4.2.* In future work this would have to be explored meticulously. Overall, the choice of the zkSNARK as the protocol aligns perfectly with reCAPTCHA's frictionless interaction by being fast and requiring no interaction from the client, satisfying the objective.

### 5.1.3 Objective 3

This objective examines the suitability of the chosen ML algorithm. While initially all the ML algorithms were considered and some implementation was attempted with the KNN, ultimately the SVM and specifically a pivot to a one class SVM was chosen. This decision was based on the fact that the dataset contained only one class. Research cited in 3.3.4, demonstrates that this model can be used for anomaly detection. There were also attempts to find different datasets that contained both human and bot data, but no such findings were discovered. Due to this, it was the most suitable of the three ML algorithms examined. While there are some limitations stemming from this choice, outlined in the next chapter, the assumptions of the project have been proven through the results of the implementation and so the objectives is rightfully satisfied.

### 5.1.4 Objective 4

This objective examines how effective the integration of a zkML into reCAPTCHA is. In terms of the effectiveness of the zkSNARK protocol itself, as it was only expanded theoretically and not practically, then a thorough security evaluation that checks whether the protocol is sound, complete and zero-knowledge as they are defined in the Results chapter cannot be examined. However, the ZKML was examined from a research perspective, its' position in relevant literature and a threat model detailing any potential attacks, their adversaries, the mitigations, the likelihood that the attack can happen and their severity if they were to be successful. Research shows that the use of a ZKML on a CAPTCHA system is quite unique, positioning this project's output in an area with a research gap.

The code of the implementation was tested on datasets the model has not seen before and compared to the main one used. The evaluation consisted of an analysis of the runtime, processing power and memory which derived that the model was slow and took a lot of memory without utilising enough of the device's CPU. This is most likely due to the author's computational limitations.

In summary, due to the fact that the $Gen$ and $Proof$ algorithms of the ZKML were not implemented as the proof of concept, the effectiveness of the integration of a ZKP into an ML for use in a CAPTCHA verification system cannot be thoroughly examined. However, the proposed ZKML framework was evaluated theoretically and still provides an idea of whether a zkSNARK into reCAPTCHA can prove effective. Hence, the objective is partially satisfied.

## 5.2 Summary

Overall, the objectives of the project, namely 1, 2 and 3, were fully satisfied with only objective 4 being partially met. The project defines what human behavioural characteristics are and derives an adaptation of a zkSNARK with a OC-SVM, that can securely verify that a web client is human without any concerns over the privacy of their data. This framework is theoretically proven to be quite effective for reCAPTCHA leaving room for improvement.

# 6 Evaluation, Reflections and Conclusions

This chapter contains an overall reflection of the project, alongside the limitations presented, future work and a conclusion.

## 6.1 Project Evaluation

### 6.1.1 Choice of objectives

All four objectives shaped the direction of this project to completion. Objectives 1, 2 and 3 were successfully satisfied through a combination of the thorough research conducted, the implementation and the selection of the appropriate dataset. That being said, Objective 4 was not realised to the extent of the author's wishes, as the proof of concept focused more on the ML implementation, making the security evaluation of the ZKP difficult to examine. This project's subjects, specifically the mathematics behind the ZKPs were quite complex and took a lot of time for understanding for the author, ultimately not leaving enough time for a more thorough expansion of the project's proof of concept. If the author was to develop this project again, they would focus heavily on the mathematics of the zkSNARK and eventually create a thorough implementation of the produced ZKML that is proven to be sound, complete and zero-knowledge.

### 6.1.2 Literature

Given the number of topics this project covers, the literature review played a crucial role. The vast amount of research of different ZKML applications provided the author with very useful information to understand how the concept works and subsequently how it can be applied to the scenario of this project.

### 6.1.3 Methods

The methods were carefully crafted and applied to ensure that each step of the project was thoroughly completed. Setting goals throughout the project on what should be expected of the results, provided the author with a roadmap, that allowed the project to progress efficiently. While there was a setback with understanding which ML to use for a one class dataset, the author used their literature skills and was able to find research papers that justified the use of a one-class SVM for anomaly detection.

### 6.1.4 Results

The results of this project provide a foundation towards a more secure reCAPTCHA. They satisfy the main three objectives of the project and the fourth was only partially evaluated and did not reach the level of depth the author initially expected. Regardless, the results of the project can serve as guidance for how to improve the security of reCAPTCHA in the future.

### 6.1.5 Limitations

1. The dataset comprises of 12 user CSV files, each user separately. Some of these files are as large as 2GB. Ideally, the author planned to merge all the files together as this would provide for a much more comprehensive analysis. However, taking in consideration that the author conducted the implementation on a personal laptop, this unfortunately restricted the capacity of processing this large-scale data. This is a computational limitation.
2. Initially the author planned to use the KNN algorithm, however upon the discovery that the dataset consisted of only one classification (unlike the two classifications that the KNN required), the approach had to be shifted. Therefore, the author ended up using a one-class SVM to effectively lead to the desired result.

### 6.1.6 Future Work

1. Future work can avoid the computational complexities of processing such large data on a personal device by leveraging cloud-based solutions, specifically utilising AWS EC2 instances.
2. The proof of concept is a conceptual implementation of the verification process. Implementing this within a reCAPTCHA system would require taking in consideration the encryption that occurs at the first two algorithms of the zkSNARK instead of using a multiplication of the new data by x as is done in this project.
3. The computations that are expected of the client-prover are clearly too complex for the client to compute but even if that was a possibility they are always seen as untrusted until they prove that they are trustworthy by passing the verification. To solve this, there needs to be a way for the prover's device or browser to be able to compute the main two algorithms. In the evaluation of the results, it is discussed that this can be done on the browser however, a significant amount of research and time needs to be put into this.
4. The ZKML is required to use a trusted authority to complete its' $Gen$ algorithm. Future work needs to focus on who exactly will act as this authority and why and how they will be transmitting the proving and verification keys to the client-prover and server-verifier.
5. The weights of the model can be committed to ensure transparency for verification from third parties to ensure integrity.

## 6.2 Reflections

Overall, this project proved to be very insightful for the author. It challenged their research capabilities and expanded their mathematical and technical knowledge. Despite how complex some concepts were, the process of discovering and learning was both enjoyable and rewarding, supporting in the author's personal and academic growth significantly.

## 6.3 Conclusions

In conclusion, this project gathers together extensive research on the integration of zero knowledge proofs with machine learning algorithms. It composes the zkSNARK protocol, with a One-Class SVM to create a unique ZKML that if applied to Google's reCAPTCHA system can ensure that the web client's data is secure and GDPR-compliant. This project takes in consideration all the research surrounding the field and constructs its own scenario for how a web client can verify that they are human to the reCAPTCHA server and implements the verification algorithm of the ZKML using Python with testing that attests to the legitimacy of the code. It proposes a unique design placed within a research gap that can lead to a more secure version of reCAPTCHA, creating the building blocks for future advancements in the field.

# 7 References

Google for Developers. (2018). *Introducing reCAPTCHA v3: the new way to stop bots | Google Search Central Blog | Google for Developers*. [online] Available at: https://developers.google.com/search/blog/2018/10/introducing-recaptcha-v3-new-way-to [Accessed 13 Sep. 2024].

Zhang, J., et al (2020). Zero Knowledge Proofs for Decision Tree Predictions and Accuracy. Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. doi:https://doi.org/10.1145/3372297.3417278.

Grant, C. (2020). Human-guided burrito bots raise questions about the future of robo-delivery. [online] Thehustle.co. Available at: https://thehustle.co/kiwibots-autonomous-food-delivery [Accessed 13 Sep. 2024].

I. Tsingenopoulos, et. al (2022) Captcha me if you can: Imitation Games with Reinforcement Learning, 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), doi: 10.1109/EuroSP53844.2022.00050.

Akrout, I., et. al (2019). Hacking Google reCAPTCHA v3 using Reinforcement Learning. [online] arXiv.org. Available at: https://arxiv.org/abs/1903.01003 [Accessed 13 Sep. 2024].

Kotaro T. (2022). Discovering Critical Weaknesses in reCAPTCHA v3.

6sense. (2024). Best CAPTCHA Software in 2024 | 6sense. [online] Available at: https://6sense.com/tech/captcha [Accessed 13 Sep. 2024].

Xu, X., et al (2020). A survey of CAPTCHA technologies to distinguish between human and computer. Neurocomputing, [online] 408, pp.292–307. doi:https://doi.org/10.1016/j.neucom.2019.08.109.

Google for Developers. (2018). Choosing the type of reCAPTCHA. [online] Available at: https://developers.google.com/recaptcha/docs/versions [Accessed 13 Sep. 2024].

Jiang, N., Dogan, H. and Tian, F. (2017). Designing Mobile Friendly CAPTCHAs: An Exploratory Study. Electronic workshops in computing. [online] doi:https://doi.org/10.14236/ewic/hci2017.92.

Feng, B., Qin, L., Zhang, Z., Ding, Y. and Chu, S. (2021). ZEN: An Optimizing Compiler for Verifiable, Zero-Knowledge Neural Network Inferences. [online] Cryptology ePrint Archive.

Available at: https://eprint.iacr.org/2021/087 [Accessed 21 Sep. 2024].

Wu, H., Zheng, W., Chiesa, A., Ada Popa, R. and Stoica, I. (2018). DIZK: a distributed zero knowledge proof system. In: SEC'18: Proceedings of the 27th USENIX Conference on Security Symposium. [online] United States: USENIX Association, pp.675–692. Available at: https://dl.acm.org/doi/10.5555/3277203.3277254 [Accessed 21 Sep. 2024].

Sanchez, et al., (2020) BEHACOM, Mendeley Data. Available at: https://data.mendeley.com/datasets/cg4br62535/2 (Accessed: 6 September 2024)

Jo, T. (2021). Machine learning foundations : supervised, unsupervised, and advanced learning. Cham: Springer.

Sun, X., F. Richard Yu, Zhang, P., Sun, Z., Xie, W. and Peng, X. (2021). A Survey on ZeroKnowledge Proof in Blockchain. IEEE Network, [online] 35(4), pp.198–205. doi:https://doi.org/10.1109/mnet.011.2000473.

Mulder, V., Alain Mermoud, Lenders, V. and Bernhard Tellenbach (2023). Trends in Data Protection and Encryption Technologies. Springer.

Bunz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P. and Maxwell, G. (2018). Bulletproofs: Short Proofs for Confidential Transactions and More. 2018 IEEE Symposium on Security and Privacy (SP). doi:https://doi.org/10.1109/sp.2018.00020.

Anca Nitulescu (2019). A Gentle Introduction to SNARKs. [online] Available at: https://www.semanticscholar.org/paper/A-Gentle-Introduction-to-SNARKsNitulescu/0c900671fc731fda31dbb3e94bd16e9e42df66ff [Accessed 20 Sep. 2024].

Hadas Zeilberger (2019). Simple Explanations of Arithmetic Circuits and Zero-Knowledge Proofs. [online] Medium. Available at: https://medium.com/web3studio/simple-explanations-of-arithmeticcircuits-and-zero-knowledge-proofs-806e59a79785 [Accessed 21 Sep. 2024].

Morais, E., Koens, T., Cees van Wijk and Koren, A. (2019). A survey on zero knowledge range proofs and applications. SN Applied Sciences, [online] 1(8). doi:https://doi.org/10.1007/s42452-019-0989-z.

Feng, B., Wang, Z., Wang, Y., Yang, S. and Ding, Y. (2024). ZENO: A Type-based Optimization Framework for Zero Knowledge Neural Network Inference. doi:https://doi.org/10.1145/3617232.3624852.

Guerar, M., Verderame, L., Migliardi, M., Palmieri, F. and Merlo, A. (2022). Gotta CAPTCHA 'Em All: A Survey of 20 Years of the Human-or-computer Dilemma. ACM Computing Surveys, 54(9), pp.1–33. doi:https://doi.org/10.1145/3477142.

Lee, S., Ko, H., Kim, J. and Oh, H. (2024). vCNN: Verifiable Convolutional Neural Network Based on zk-SNARKs. IEEE Transactions on Dependable and Secure Computing, [online] pp.1–17. doi:https://doi.org/10.1109/tdsc.2023.3348760.

Liu, T., Xie, X. and Zhang, Y. (2021). zkCNN: Zero Knowledge Proofs for Convolutional Neural Network Predictions and Accuracy. Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. doi:https://doi.org/10.1145/3460120.3485379.

El-Hajj, M. and Oude Roelink, Bjorn (2024). Evaluating the efficiency of zk-SNARK, zk-STARK, and bulletproof in real-world scenarios: A benchmark study. Information, [online] 15. doi:https://doi.org/10.3390/info15080463.

Amer, M., Goldstein, M., Goldstein@dfki, M., De, S. and Abdennadher (n.d.). Enhancing One-class Support Vector Machines for Unsupervised Anomaly Detection. [online]

David M.J. Tax and Robert P.W. Duin (2003). Support Vector Data Description. Machine Learning, [online] 54(1), pp.45–66. doi:https://doi.org/10.1023/b:mach.0000008084.60811.49.

Ghimire, M. (2023). Google reCAPTCHA V3: How reCAPTCHA V3 Works? [2023]. [online] Privyr Blog. Available at: https://www.privyr.com/blog/google-recaptchav3/#:~:text=Once%20the%20user%20interaction%20is,user%20is%20human%20or%20not. [Accessed 30 Sep. 2024].

Raychev, J.P. (2022). Introduction and Python implementation of Google reCaptcha. [online] Medium. Available at: https://medium.com/geekculture/introduction-and-python-implementation-ofgoogle-recaptcha-e50a92c2e56f [Accessed 30 Sep. 2024].

Shrestha, P. (2024). Tuning SVM Hyperparameters: Making Your Classifier Shine Like a Pro! [online] Medium. Available at: https://medium.com/@prayushshrestha89/tuning-svmhyperparameters-making-your-classifier-shine-like-a-pro-8673639ddb16 [Accessed 30 Sep. 2024].

Medium. Available at: https://medium.com/grabngoinfo/one-class-svm-for-anomaly-detection6c97fdd6d8af [Accessed 30 Sep. 2024].

Kun-Lun Li, Hou-Kuan Huang, Sheng-Feng Tian and Wei Xu (2004). Improving one-class SVM for anomaly detection. [online] doi:https://doi.org/10.1109/icmlc.2003.1260106.

Iñigo Querejeta-Azurmendi, Papadopoulos, P., Matteo Varvello, Nappa, A., Zhang, J. and Livshits, B. (2021). ZKSENSE: A Friction-less Privacy-Preserving Human Attestation Mechanism for Mobile Devices. Proceedings on Privacy Enhancing Technologies, [online] 2021(4), pp.6–29. doi:https://doi.org/10.2478/popets-2021-0058.

Feng, B., Wang, Z., Wang, Y., Yang, S. and Ding, Y. (2024). ZENO: A Type-based Optimization Framework for Zero Knowledge Neural Network Inference. doi:https://doi.org/10.1145/3617232.3624852.

Zhang, J., Fang, Z., Zhang, Y. and Song, D. (2020) *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. doi: 10.1145/3372297.3417278.

Weng, C., Yang, K., Xie, X., Katz, J. and Wang, X. (2021). *Mystique: Efficient Conversions for ZeroKnowledge Proofs with Applications to Machine Learning*. [online] Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2021/730 [Accessed 2 Oct. 2024].

Wang, H. and Hoang, T. (2023). *ezDPS: An Efficient and Zero-Knowledge Machine Learning Inference Pipeline*. [online] Proceedings on Privacy Enhancing Technologies. Available at: https://petsymposium.org/popets/2023/popets-2023-0061.php [Accessed 2 Oct. 2024].

Xing, Z., Zhang, Z., Liu, J., Zhang, Z., Li, M., Zhu, L. and Russello, G. (2023). *Zero-knowledge Proof Meets Machine Learning in Verifiability: A Survey*. [online] arXiv.org. Available at: https://arxiv.org/abs/2310.14848 [Accessed 3 Jul. 2024].

Abla Smahi, Li, H., Han, W., Ahmed Ameen Fateh and Ching Chuen Chan (2024). VFL-Chain: Bulletproofing Federated Learning in the V2X environments. Future generation computer systems. [online] doi:https://doi.org/10.1016/j.future.2024.02.012

Li, K.-L., Huang, H.-K., Tian, S.-F. and Xu, W. (2003). Improving one-class SVM for anomaly detection. pp.3077–3081 Vol.5. doi:https://doi.org/10.1109/ICMLC.2003.1260106.

Christ, M., Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Deepak Maram, Roy, A. and Wang, J. (2024). *SoK: Zero-Knowledge Range Proofs*. [online] Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2024/430 [Accessed 2 Oct. 2024].

Dinh, N.T. and Hoang, V.T. (2023). Recent advances of Captcha security analysis: a short literature review. *Procedia Computer Science*, 218, pp.2550–2562. doi:https://doi.org/10.1016/j.procs.2023.01.229.

Ritthichai, B. (2020). *An Overview of Google reCAPTCHA and the Differences Between v2 and v3*. [online] Medium. Available at: https://medium.com/@ben.ritthichai/an-overview-of-google-recaptcha-and-the-differences-between-v2-and-v3-efcced66c1f1 [Accessed 7 Oct. 2024].

Eikvil, L. (1993). *OCR Optical Character Recognition*. [online] Available at: https://home.nr.no/~eikvil/OCR.pdf.

Sukhani, K., Sawant, S., Maniar, S. and Pawar, R. (2021). Automating the bypass of image-based CAPTCHA and assessing security. pp.01–08. doi:https://doi.org/10.1109/ICCCNT51525.2021.9580020.

Swisher, J. (2024). *CAPTCHA vs reCAPTCHA vs Akismet: Which is Best?* [online] Akismet. Available at: https://akismet.com/blog/akismet-vs-captcha-vs-recaptcha/ [Accessed 7 Oct. 2024].

Yampolskiy, R.V. (2012). AI-Complete CAPTCHAs as Zero Knowledge Proofs of Access to an Artificially Intelligent System. *ISRN Artificial Intelligence*, 2012, pp.1–6. doi:https://doi.org/10.5402/2012/271878.

scikit-learn. (2024). *MinMaxScaler*. [online] Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.preprocessing.MinMaxScaler.html [Accessed 15 Oct. 2024].

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J. and Platt, J. (1999). Support vector method for novelty detection. Denver, CO: MIT Press, pp.582–588.

Fragoso, V., Scheirer, W., Hespanha, J. and Turk, M. (2016). *One-Class Slab Support Vector Machine*. [online] arXiv.org. Available at: https://arxiv.org/abs/1608.01026 [Accessed 15 Oct. 2024].

Grassi, P.A., Garcia, M.E. and Fenton, J.L. (2017). Digital identity guidelines: *NIST Special Publication 800-63-3*, [online] p.47. doi:https://doi.org/10.6028/nist.sp.800-63-3.

Pedersen, T.P. (1992). Non-interactive and information-theoretic secure verifiable secret sharing. In: J. Feigenbaum, ed. Springer Berlin Heidelberg, pp.129–140.

Stephens-Davidowitz, N. (2023). *Zero-knowledge proofs for all of NP 1 Recap*. [online] Available at: https://www.noahsd.com/crypto_lecture_notes/CS4830_Lecture_20___ZKP_for_all_of_NP.pdf [Accessed 16 Oct. 2024].

Green, M. (2017). Zero Knowledge Proofs: An illustrated primer, Part 2. [online] A Few Thoughts on Cryptographic Engineering. Available at: https://blog.cryptographyengineering.com/2017/01/21/zero-knowledge-proofs-an-illustrated-primer-part-2/# [Accessed 16 Oct. 2024].

Appendix A

**Project Proposal:**

**Zero Knowledge Proofs for new reCAPTCHA**

Name: Katerina Giagkoumpova

Project Supervisor: Nikos Komninos

# 1 Introduction

### 1.1 Research Question

How can zero-knowledge proofs aid in the development of a secure reCAPTCHA v4?

### **1.2** Problem to be solved

Google's most recent reCAPTCHA V3 is a user friendly, frictionless verification process pushing away the disruptive past versions of clicking on images and ticking boxes. However, its reliance on a machine learning (ML) model, exposes it to vulnerabilities as stated by Tsingenopoulos et al. (2022) noting that adversaries can construct intricate models that mimic human web browsing behaviour, undermining its effectiveness.

This issue with machine learning underscores a broader challenge highlighted by Zhang et al, (2020) regarding the reproducibility and integrity of ML models. There is a lack of trust and transparency as the training data and process may not be published therefore, results cannot be reproduced.

Due to this, Kotaro (2022) discovered four novel findings that highlight the issue of bots bypassing the verification. reCAPTCHA v3 employs a JavaScript API to provide a score ranging from 0 to 1.0, 0 indicating high likelihood of bot activity while 1.0 suggests high likelihood of human interaction.

1. The default score provided by v3 is 0.9 out of 1.0 and any deviations from the expected human behaviour decrease it.
2. A significant volume of requests, exceeding 500 consecutive ones, identifies web crawlers as bots only on the subsequent day.
3. A sleep duration of ten seconds between requests does not flag web crawlers as bots.
4. And finally, requests with matching language codes in the "accept-language" header and the IP address location, with dynamic user behaviour like keyboard typing and basic human like movements could possibly evade reCAPTCHA v3's bot detection mechanism.

### **1.3** Aims and Objectives

The main aim of this project is to explore how the integration of non-interactive ZKPs into the domain of ML models can address the vulnerabilities present in reCAPTCHA v3.

The objectives of the project are stated below:

1. Identify what characteristics determine human web browsing behaviour.
2. Identify which parameters of non-interactive ZKPs can be implemented into CAPTCHA.
3. Identify which characteristics of ML algorithms can be implemented into CAPTCHA.
4. Evaluate how effective the integration of ZKPs and ML models (ZKML) is when implemented into CAPTCHA.

### **1.4** Output

The output is the development of a novel Zero-Knowledge Machine Learning (ZKML) Proof that allows the reCAPTCHA v3 to run as required while at the same time concealing the details about the ML model being used.

### **1.5** Beneficiaries

#### 1.5.1 Google

Google can benefit the most from this as this will safeguard their ML data while proving its' accuracy and security.

#### 1.5.2 End users

ReCAPTCHA is a widely used service so increased protection from bots will safeguard many users from the malicious activities that bots may perform.

#### 1.5.3 Website owners

As reCAPTCHA is globally used, website owners will have a safer method of verifying humans and forbidding entry to bots.

#### 1.5.4 Researchers/Academics

The output of this project and the findings discovered, may contribute to breaching the current vulnerabilities in reCAPTCHA v3.

# 2 Critical Context

### **2.1** CAPTCHA versions and their challenges

ReCAPTCHA Inc, is a CAPTCHA system owned by Google that enables web hosts to distinguish between human and automated access (bots) to websites. It consists of three versions, with the most popularly used, version 3. Version 1, now discontinued, asked users to identify distorted letters with the idea that human eyes and memory of reading will aid humans but will be difficult for bots to bypass. Version 2 introduced image-based challenges, however was facing accessibility issues (Khalifa et al., 2016). Due to deep learning algorithms and the usage of OCR (Optical Character Recognition) techniques, bots have been able to bypass these verification methods and mimic human behaviour (Kotaro 2022). Currently, v2 has been mostly replaced by v3.

Volume 3 of CAPTCHA, reCAPTCHA, introduced a frictionless experience where the user does not have to interact with the system. The application tracks the user's actions on the website and based where the user score is located with respect to the threshold, it determines whether they are human or not (Google for Developers, 2018). This makes this version the most user friendly yet as the user itself does not have to tick any boxes or complete any challenges.

## **2.2** Machine Learning Models

While Google has not disclosed whether they utilise ML models for v3, Google for Developers (2018) points out that reCAPTCHA uses an advanced risk analysis engine and adaptive challenges, a potential indication of the usage of a combination of machine learning models. Mohri et al. (2018) define machine learning (ML) as computational techniques that utilise data created for analysis to enhance the quality of the methods or to precisely predict results.

The types of ML algorithms are portrayed in Table 1, with data taken from Jo (2021).

| ML Type | How it works | Training set | Applications |
|---|---|---|---|
| Supervised | Takes inputs with corresponding outputs to make predictions on new data. | Labelled data | Classifications<br>Regressions |
| Unsupervised | Takes inputs with unknown outputs and identifies trends in the data. | Unlabelled data | Clustering<br>Dimensionality<br>Reduction |

| | | | |
|---|---|---|---|
| Semi-Supervised | Uses a supervised model for labelled data and an unsupervised model for unlabelled data. | Both labelled and unlabelled data | Naïve Bayes<br>Low Density Separation |
| Reinforcement Learning | “An agent learns to act in an environment by carrying out actions and observing the results" (Sheikh Amir Fayaz et al., 2022) | No training set, the agent learns through trial and error. | Positive RL Negative RL |
| Deep Learning | Based on the model its trained on. | “Many classifiers working together, based on linear regression and some activation functions.” (Dong et al. 2021) | Convolution neural network<br>Natural language processing |

*Table 1: Types of ML algorithms*

In particular, supervised algorithms comprise of logistic regression, decision trees, naïve bayes, support vector machine (SVM) and supervised neural networks and supervised knearest neighbour.

As mentioned in 1.2, the issues with keeping the training set of the ML models secret, besides the lack of transparency is also the potential loss of verification by other parties. In addition, services may be fraudulent in claiming they use specific models.

## 2.3 Zero-Knowledge Proofs

Zhang et al. (2020)’s suggestion of the use of ZKPs into the domain of the ML model potentially addresses the problems of ML models. This approach could allow model owners to verify the integrity and accuracy of their model without revealing the entire model itself, protecting their intellectual property and by publicising it, allowing transparency and potentially the model’s reproducibility.

Zero knowledge proofs (ZKP) are cryptographic protocols that allow one party (the prover) to demonstrate to another party (the verifier) knowledge of a secret without revealing the secret itself. This ensures that the prover can prove to the verifier their knowledge without compromising the confidentiality of the information being proved.

For a ZKP to be applied successfully, its’ requirements need to be met.

Completeness: If the prover knows the secret the verifier will always accept the prover’s answer granted, they both follow the protocol.

Soundness: if the prover does not know the secret, then the verifier will always reject the prover's answer.

Zero Knowledge: the verifier only knows that the statement is correct and nothing about the statement itself.

This project will focus on non-interactive ZKPs.

### 2.4 Non-interactive Zero Knowledge Proofs

**Non-interactive** Zero-Knowledge Proofs is a type of ZKPs that allows a prover to demonstrate knowledge of a secret without the need for interaction with the verifier. These characteristics are suitable for web browsing verification as minimal trust is placed on the web client until they prove they are a human. Thus, constant interaction between the prover and verifier would not be secure. This also aligns with the seamless experience of CAPTCHA v3 where the user is not bothered by different challenges they have to complete. The description of the potential ZKPs that will be tested are described alongside their characteristics and why they are suitable for application within reCAPTCHA.

- Bulletproofs do not require a trusted setup Morais et al. (2019). They can validate a range of values while being faster than traditional ZK range proofs. This is perfect for the implementation in captcha where the user (prover) is not trusted.
- zkSNARKS by definition are succinct – the proof's size is significantly smaller in comparison to size of the computation itself according to Nitulescu, (2020). This means that the verification is fast and aligns perfectly with the need of verifying a large amount of users that enter a popular web page, achieving efficiency and speed.
- In Zero-Knowledge Set Memberships (ZKSM) by definition the prover wants to demonstrate to the verifier than an element X belongs in a set S without revealing any information about S itself. This aligns with the proposed implementation of SVM, where depending on whether the user's score (X) in the similarity metric, belongs to a predetermined set S classified as human web behaviour, the user is verified as human. If the score X does not belong in S, the user is flagged as a bot.

## 3 Approaches: Methods & Tools for Design, Analysis & Evaluation

### 3.1 Environment/ Specification requirements

I will be developing this project on my laptop using Python. The specification requirements that define what human web browsing behaviour are preferably characteristics such as keystrokes, mouse movement, context of pages visited, and applications accessed.

These requirements are found in a dataset from Sanchez Sanchez et al. (2020), comprising of the web behaviour of 12 anonymous users over the course of 55 days, with each user having their own dataset file. With approximately 16000 rows, there is a good amount of data for training the model. The dataset will need to be prepared by pre-processing such as merging all the user data into one, cleaning it up and ensuring the target variable is clearly labelled as human.

## 3.2 Design

The design of the project can be separated into three main stages:

By analysing existing ZKP schemes through an extensive literature review I aim to identify those with properties that are most desired with implement in CAPTCHA. At this stage in my research, the most favourable schemes seem to be the zkSNARKS, Bulletproofs and Zero-Knowledge Set Membership proofs for the reasonings provided in section 2.4. I will then design the ZKP scheme with foundations from the methodology in Morais et al. (2019) which constructs Zero-Knowledge Range Proofs including Bulletproofs.

Based on this database, I will investigate different ML algorithms that are suitable in the implementation of reCAPTCHA. While reinforcement learning is a possibility, I will start with supervised learning:

**SVM algorithm:** The SVM's utilisation of kernel functions and vector comparisons in a feature space could potentially enable the efficient measure of the similarities between the training data (human browsing behaviour dataset) and new data (from user entering a page).

**K-nearest neighbour (KNN):** The algorithm, as stated by Uddin et al., (2022), comprises of a variable parameter 'k', to identify the number of nearest neighbours of a new entry point. Can be applied in reCAPTCHA in similar fashion to SVM.

**Recurring neural networks (RNN)**: "Information cycles through a loop. It takes both the current input and also what it has learned from the inputs it received previously" (Kanagachidambaresan et al., 2021) This happens by adjusting neuron weights, making it more efficient and smarter over time.

Design of ZKML:

The design of the integration of the ZKPs and the ML algorithm needs to be considered here. The approach will be based off of the ZKML model introduced by Chen et al, (2024) where the ML models are translated into ZKP circuits.

### 3.3 Development & Implementation

Based on the design on the ZKML, I will start the implementation. The ML algorithm will be converted to ZKP circuits, establishing a rulebook that ensures the ML algorithm functions correctly within the chosen ZKP method as according to Veridise (2024).

As in Chen et al. (2024) the ZKML is split into two parts: the gadgets and the optimiser. Gadgets are circuits performing specific operations within the ML model based on the algorithm used. The research article notes some gadgets such as dot products – used for similarity of data in SVM and RNN and softmax functions used in the hidden layer of neural networks (RNN). The circuit optimiser optimises the arrangement of these gadgets in the circuit. These approaches will be implemented in this project, but the choice of gadgets will be based on the particular ML algorithm chosen.

### 3.4 Analysis & evaluation

Testing: The ZKML will be integrated into a CAPTCHA-like demonstration to evaluate its success. In the proposed demo, there will be a numeric threshold; users who meet the threshold are verified as humans, while those who do not are flagged as bots. For example, for a SVM algorithm, a threshold will be set on the margin. If new data falls within this threshold, then this indicates there is high similarity with human browsing behaviour therefore, the new user will be considered human but if it falls outside the range, it will be classified as bot.

I will have to conduct the following:

1. An analysis to determine whether the derived ZKML retains the ZKP properties of completeness, soundness and zero-knowledge
2. An evaluation of the effectiveness of the new ZKML's gadgets and optimiser by tuning their parameters: I will modify aspects of the model and examine how they will affect the model and by how much. For the three proposed algorithms:
    - SVM: How will the classification be affected if the kernel function, responsible for the projection of the data points, is altered
    - RNN: If allowed by the training set, check how a different activation function could change the effectiveness of the model to learn complex patterns.
    - KNN- If k parameter is altered becoming more sensitive to outliers, how does this affect the distance metric.
3. Determine the practicality vs the privacy trade-off.

## 4 Work Plan

Figure 1 portrays the Gantt chart organising the project into weeks starting from 10/06/2024 to 30/09/2024 as 02/10 is the submission deadline.

A lot of tasks overlap with each other however, this is not an issue as I will ensure each week is organised correctly and the work gets completed on time. I have organised a week where I can make up for lost time if needed in case anything goes wrong.

The plan is to finish the project with 4 weeks left until submission, 3 weeks to write the draft and one to write the final report. Steps will be reported throughout all the steps of the project.

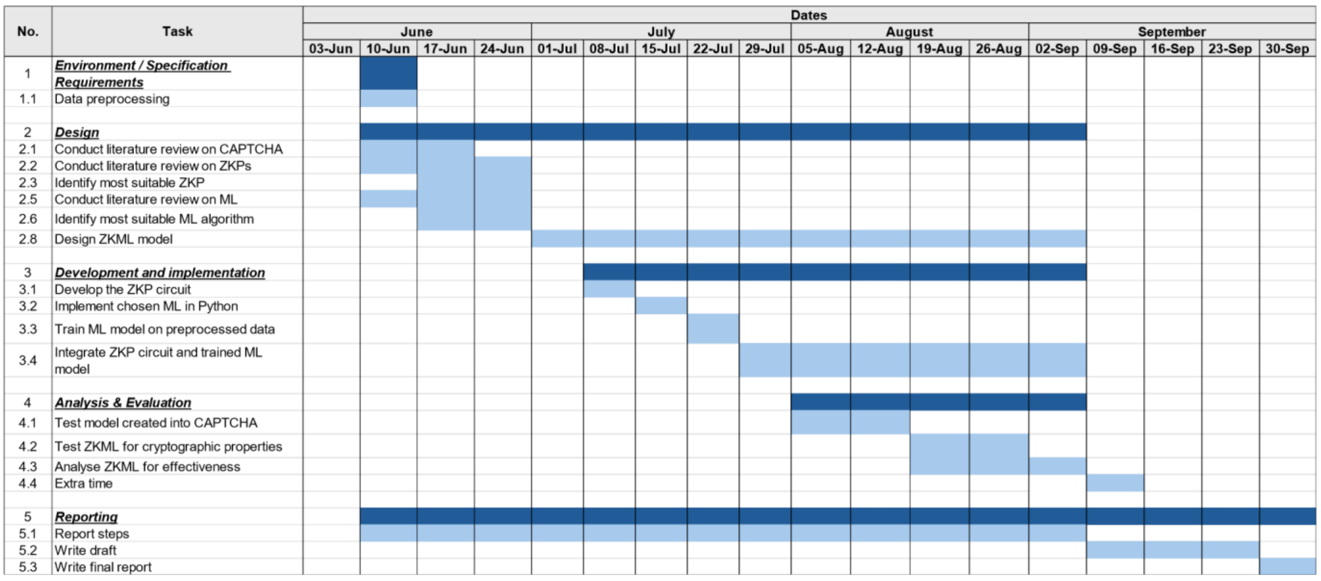


*Figure 1: Gantt chart*

## 5 Ethical review

The Ethical Review Checklist is located at the end of the document and states that there will be no participants involved in this project.

## 6 Risks

The following table, Table 2, displays some of the potential risks that may appear throughout the course of the dissertation. The scale ranges from 1 to 10 with 1 being low and 10 being high.

| Risk description | Likelihood | Severity | Mitigation |
|---|---|---|---|

| | | | |
|---|---|---|---|
| The ZKP scheme and/or the ML model that I choose in the design step will not be suitable by the time I get to the implementation. | 2 | 10 | I will contact my supervisor as soon as I find any issues and will refer to the literature review to find alternative schemes/algorithms that could work instead. For the time that may be lost, I have allocated an extra week in the Gantt chart to catch up just in case. |
| The integration of ZKP with ML may not work – problems in the code. | 4 | 8 | I will be very careful when writing the code and will conduct thorough research beforehand. I will contact my supervisor as soon as I run into a problem. |
| Failure to produce results – cannot get valid measurements of effectiveness. | 4 | 10 | Contact supervisor as soon as possible to identify the mistake that could be causing this, and if necessary, the diversion I may need to take. |
| Dataset may not contain suitable data – e.g. too many missing values | 2 | 6 | Check if the preprocessing was done correctly and double check with the supervisor regarding the possibility of finding a new dataset. |
| Hardware crash – data is lost | 3 | 6 | All the resources, notes and code will be backed up regularly to the university OneDrive account. |
| Lack of time to write report | 2 | 7 | I will be taking notes throughout the process and ensure that I meet each weekly deadline so I can remain on time and write the report at the allocated time I have given myself. |

## 7 References

[1] Ilias Tsingenopoulos, Davy Preuveneers,

Desmet, L. and Wouter Joosen (2022). Captcha

Me If You can: Imitation Games with Reinforcement Learning. [online]

doi:https://doi.org/10.1109/eurosp53844.2022.00 050. [2] Zhang, J.,

Fang, Z., Zhang, Y. and Song, D.

(2020). Zero Knowledge Proofs for Decision Tree Predictions and Accuracy. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. doi:https://doi.org/10.1145/3372297.3417278.

[3] Kotaro T. (2022). *Discovering Critical Weaknesses in reCAPTCHA v3*.

[4] Google for Developers. (2018). *Introducing reCAPTCHA v3: the new way to stop bots | Google Search Central Blog | Google for Developers*. [online] Available at: https://developers.google.com/search/blog/2018/ 10/introducing-recaptcha-v3-new-way-to [Accessed 19 May 2024].

[5] Chen, B.-J., Suppakit Waiwitlikhit, Stoica, I. and Kang, D. (2024). ZKML. doi:https://doi.org/10.1145/3627703.3650088.

[6] Mohri, M., Rostamizadeh, A. and Talwalkar, A. (2018). *Foundations of Machine Learning*. 2nd ed. Cambridge, Massachusetts: The Mit Press.

[7] Jo, T. (2021). Machine Learning Foundations. *SpringerLink*. [online] doi:https://doi.org/10.1007978-3-030-65900-4.

[8] Sheikh Amir Fayaz, S Jahangeer Sidiq, Zaman, M. and Muheet Ahmed Butt (2022a). Machine Learning: An Introduction to Reinforcement Learning. [online] pp.1–22. doi:https://doi.org/10.1002/9781119776499.ch1.

[9] Dong, S., Wang, P. and Abbas, K. (2021). A survey on deep learning and its applications. *Computer science review*, [online] 40, pp.100379–100379.

doi:https://doi.org/10.1016/j.cosrev.2021.100379

.

[10] Morais, E., Koens, T., Cees van Wijk and Koren, A. (2019). A survey on zero knowledge range proofs and applications. *SN applied sciences/SN Applied Sciences*, [online] 1(8). doi:https://doi.org/10.1007/s42452-019-0989-z.

[11] Nitulescu, A. (2020). *zk-SNARKs: A Gentle Introduction*. [online] Available at:

https://www.di.ens.fr/~nitulesc/files/SurveySNARKs.pdf.

[12] Sanchez Sanchez, P.M., et al. (2020). BEHACOM. *data.mendeley.com*, [online] 2.

doi:https://doi.org/10.17632/cg4br62535.2.

[13] Uddin, S., Haque, I., Lu, H., Moni, M.A. and Gide, E. (2022). Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction. *Scientific Reports*, 12(1). doi:https://doi.org/10.1038/s41598-022-10358-x.

[14] Veridise (2024b). *Zero Knowledge for Dummies: Demystifying ZK Circuits*. [online] Medium. Available at: https://medium.com/veridise/zero-knowledgefor-dummies-demystifying-zkcircuitsc140a64c6ed3 [Accessed 15 May 2024].

[15] Kanagachidambaresan, G.R., Adarsha
Ruwali, Banerjee, D. and Kolla Bhanu Prakash
(2021). Recurrent Neural Network. *EAI/Springer Innovations in Communication and Computing*, [online] pp.53–61. doi:https://doi.org/10.1007/978-3-030-570774_7.

[16] Mahesh, B. (2019). *Machine Learning Algorithms -A Review*. [online] ResearchGate. Available at:
https://www.researchgate.net/publication/34471 7762_Machine_Learning_Algorithms_A_Review [Accessed 19 May 2024].

[17] Khalifa, W. and Hasan, A. (2016). A Survey of Current Research on CAPTCHA. *International Journal of Computer Science & Engineering Survey (IJCSES)*, 7(3).

## Research Ethics Review Form: BSc, MSc and MA Projects

## Computer Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

***PART A: Ethics Checklist***. All students must complete this part. The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

***PART B: Ethics Proportionate Review Form***. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be ***provisional*** – *identifying the planned research as* likely to involve MINIMAL RISK. In such cases you must additionally seek ***full approval*** from the supervisor as the project progresses and details are established. ***Full approval*** must be acquired in writing, before beginning the planned research.

| | | |
|---|---|---|
| **A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/** | | *Delete as appropriate* |
| 1.1 | Does your research require approval from the National Research Ethics Service (NRES)?<br>*e.g. because you are recruiting current NHS patients or staff?*<br>*If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/* | **NO** |
| 1.2 | Will you recruit participants who fall under the auspices of the Mental Capacity Act?<br>*Such research needs to be approved by an external ethics committee such as NRES or the Social*<br>*Care Research Ethics Committee - http://www.scie.org.uk/research/ethics-committee/* | **NO** |
| 1.3 | Will you recruit any participants who are currently under the auspices of the Criminal Jus System, for example, but not limited to, people on remand, prisoners and those on probation?<br>*Such research needs to be authorised by the ethics approval system of the National Offender Management Service.* | t **NO** |

| | | |
|---|---|---|
| **A.2 If you answer YES to any of the questions in this block, then unless you are applying an external ethics committee, you must apply for approval from the Senate Research Et**<br><br>**Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/** | | **h**<br><br>*Delete as appropriate* |
| 2.1 | Does your research involve participants who are unable to give informed consent?<br>*For example, but not limited to, people who may have a degree of learning disability or mental heal problem, that means they are unable to make an informed decision on their own behalf.* | **NO** |
| 2.2 | Is there a risk that your research might lead to disclosures from participants concerning t involvement in illegal activities? | h **NO** |
| 2.3 | Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)? | **NO** |
| 2.4 | Does your project involve participants disclosing information about special category or sensitive subjects?<br>*For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings* | **NO** |
| 2.5 | Does your research involve you travelling to another country outside of the UK, where th Foreign & Commonwealth Office has issued a travel warning that affects the area in whic you will study?<br>*Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/* | e **NO** |
| 2.6 | Does your research involve invasive or intrusive procedures?<br>*These may include, but are not limited to, electrical stimulation, heat, cold or bruising.* | **NO** |
| 2.7 | Does your research involve animals? | **NO** |
| 2.8 | Does your research involve the administration of drugs, placebos or other substances to study participants? | **NO** |

| | | |
|---|---|---|
| **A.3 If you answer YES to any of the questions in this block, then unless you are applying an external ethics committee or the SREC, you must apply for approval from the Comput Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/** <br> **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.** | | *Delete as appropriate* |
| 3.1 | Does your research involve participants who are under the age of 18? | **NO** |
| 3.2 | Does your research involve adults who are vulnerable because of their social, psychologi or medical circumstances (vulnerable adults)? <br> *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.* | **NO** |
| 3.3 | Are participants recruited because they are staff or students of City, University of London <br> *For example, students studying on a particular course or module.* <br> *If yes, then approval is also required from the Head of Department or Programme Director.* | **NO** |
| 3.4 | Does your research involve intentional deception of participants? | **NO** |
| 3.5 | Does your research involve participants taking part without their informed consent? | **NO** |
| 3.5 | Is the risk posed to participants greater than that in normal working life? | **NO** |
| 3.7 | Is the risk posed to you, the researcher(s), greater than that in normal working life? | **NO** |
| **A.4 If you answer YES to the following question and your answers to all other questions sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK.** <br> **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** <br> **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.** | | *Delete as appropriate* |
| 4 | Does your project involve human participants or their identifiable personal data? <br> *For example, as interviewees, respondents to a survey or participants in testing.* | **NO** |

# Appendix B

This is the code that produces the outputs of *Chapter 4 Results 4.3.2.* For the testing section of the Results chapter, the same code is implemented with the only difference being the dataset that is loaded. For user 9, the dataset “User9_BEHACOM.csv” is loaded and for user 10 it is “User10_BEHACOM.csv”. These are available in the submission files.

---

# Loading all necessary libraries

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.svm import OneClassSVM

from scipy.spatial.distance import cdist

from sklearn.preprocessing import MinMaxScaler

## Dataset details

"This dataset contains the behavioural data of 11 different users freely utilising their personal computers" (Sanchez 2020). In this project, the data of only 1 user is taken into consideration. This data contains 10114 rows and 12048 columns, making it a large enough dataset to consider on it's own.

## Data Preprocessing

In this section:

- <u>The dataset is loaded:</u>

The dataset is in the form of a CSV file, titled as *User4_BEHACOM.csv*, and is loaded into a pandas DataFrame using the pandas library with a *latin1* encoding to handle special characters.

- <u>Unnecessary columns are removed:</u>

  - *'current_app'* and *'penultimate_app'*:

  These columns don't contain numeric values, which can cause issues in calculations later on.

- *'USER'*:

This column indicates which user out of the 11 from the study the dataset belongs to. Since it's part of a larger dataset and is not needed for the analysis in this project, it is considered unnecessary.

- *'timestamp'*:

This column denotes the timestamps that the entries were made in. This is not deemed as essential information, as it does not directly reflect a behavioural characteristic. An alternative approach would be to set this column as the index, which can help with analysing how fast the user makes decisions in comparison to a bot.

```
# Loading the dataset

User4 = pd.read_csv('User4_BEHACOM.csv', encoding='latin1')

# Removing unnecessary columns

B = User4.drop(columns=['current_app', 'penultimate_app','USER', 'timestamp'])

print(B)
```

## Data Preparation

In this section:

- <u>The training data is defined:</u>

The variable *B_train* is assigned to the DataFrae *B*, making up the training set.

- <u>The testing data is defined:</u>

The variable *B_test* is also assigned to *B* and multiplied by a scalar *x*. This scalar represents the privacy preservation property of the project's output. For more information see *Chapter 3 Methods*.

- <u>Both variables are scaled in a MinMaxScaler Instance:</u>

This will scale deatures to a range between 0 and 1. Both the training and testing sets scale their values between 0 and 1 with the testing being slightly larger due to the multiplication with scalar *x*. This is done in reference to [2].

```
# Defining training set

B_train = B
```

```
# Random scalar

x = 50


# Defining testing data

B_test = B * x

scaler = MinMaxScaler()


# The training data is fitted and transformed

B_train_scaled = scaler.fit_transform(B_train)


# The testing data is transformed

B_test_scaled = scaler.transform(B_test)


print(B_test_scaled)

print(B_train_scaled)
```

## One-class SVM and the decision boundary

In this section:

- <u> The one-class(OC) SVM model is created:</u>

The oc SVM is initialised with the suitable hyperparameters. The kernel used is the *rbf*, the kernel coefficient, *gamma*, is on the automatic setting and *nu* is a small value. This is done with reference to [3].

- <u> The model is fitted to the training set</u>

- <u> The decision function is used:</u>

The decision function computes the distance of each point in the data from the decision boundary, with positive values denoting data within the boundary and negative values, the points outside the boundary. This is done in reference to [3]]. With this decision function, the minimum and maximum values are derived.

- <u> The new minimum value is derived:</u>

As the minimum value turns out to be negative, this indicates an outlier. Using a *for loop* the next value after the minimum that is of a positive sign becomes the new minimum value of the output of the decision function.

# Visualisation

With the use of the seaborn library and in reference to [4], the following are visualised:

- The outputs of the decision functions

- The negative minimum point and the new minimum point as dots.

```
model = OneClassSVM(kernel='rbf', gamma='scale', nu=0.1)
model.fit(B_train_scaled)
decisionf = model.decision_function(B_train_scaled)
# Minimum value
minvalue = decisionf.min()
# Maximum value
maxvalue = decisionf.max()


print(f'The minimum and maximum values are', '[', minvalue, maxvalue, ']')
sns.set(style='whitegrid', palette='muted')


plt.figure(figsize=(10, 6))


# Plotting the decision function output data
sns.scatterplot(x=range(len(decisionf)),
                y=decisionf,
            color='purple',

            marker='o',

            s=30,

            label='Decision Function Values',

            alpha=0.6)


plt.title('User 4 Decision Function Values', fontsize=14)

plt.xlabel('Index', fontsize=14)

plt.ylabel('Decision Function Value', fontsize=14)


plt.grid(True)

plt.legend()


plt.show()

sns.set(style='whitegrid', palette='muted')


plt.figure(figsize=(10, 6))


# Plotting the decision function output data

sns.scatterplot(x=range(len(decisionf)),

            y=decisionf,

            color='purple',

            marker='o',

            s=30,

            label='Decision Function Values',

            alpha=0.6)
```

```
plt.title('User 4 Decision Function Positive Values', fontsize=14)

plt.xlabel('Index', fontsize=14)

plt.ylabel('Decision Function Value', fontsize=14)


plt.ylim(0, 54)

plt.grid(True)

plt.legend()


plt.show()
```

The idea behind the use of the decision function is to define the minimum and maximum values of the decision function. This is done so that there is a reference point to compare the distances between the training and testing sets.

Hence, the minimum and maximum values are revealed.

The minimum value derived from the decision function is negative, suggesting the presence of a potential outlier, which is undesirable. To address this, the first positive point encountered after this minimum becomes the new minimum value of the boundary.

```
# Creating new variable newmin

newmin = None


# The decision values are sorted in ascending order and iterated over each element.

for value in sorted(decisionf):

    # Finding the next positive value after the minimum

    if value > minvalue and value > 0:

        # Assigns the new minimum to the previously created variable

        newmin = value

        break

print(newmin)
```

```
sns.set(style='whitegrid', palette='muted')


plt.figure(figsize=(10, 6))


# Plotting the decision function output data
sns.scatterplot(x=range(len(decisionf)),
            y=decisionf,
            color='purple',
            marker='o',
            s=30,
            label='Decision Function Values',
            alpha=0.6)


# Index of the minimum value
minindex = np.argmin(decisionf)


# Index of maximum value
maxindex = np.argmax(decisionf)


# Plotting the minimum value
sns.scatterplot(x=[minindex],
            y=[minvalue],
            color='red',
            s=100,
            edgecolor='lightgrey',
            linewidth=1.5,
            label=f'Minimum: {minvalue}')
```

```
# Plotting the maximum value
sns.scatterplot(x=[maxindex],
                y=[maxvalue],
                color='green',
                s=100,
                edgecolor='lightgrey',
                linewidth=1.5,
                label=f'Maximum: {maxvalue}')

# Plotting horizontal lines at the minimum and maximum
plt.axhline(y=minvalue,
            color='red',
            linestyle='--',
            label='Boundary')

plt.axhline(y=maxvalue,
            color='red',
            linestyle='--')

# Adding shading between the minimum and maximum
plt.fill_between(range(len(decisionf)),
                 minvalue,
                 maxvalue,
                 color='lightblue',
                 alpha=0.5)
```

```
plt.title('Decision Function Range', fontsize=14)
plt.xlabel('Index', fontsize=14)
plt.ylabel('Decision Function Value', fontsize=14)

plt.grid(True)
plt.legend(loc='upper left')

plt.show()
sns.set(style='whitegrid', palette='muted')

plt.figure(figsize=(10, 6))

# Plotting the decision function output data
sns.scatterplot(x=range(len(decisionf)),
            y=decisionf,
            color='purple',
            marker='o',
            s=30,
            label='Decision Function Values',
            alpha=0.6)

# Index of the minimum value
min_index = np.argmin(decisionf)

# Plotting the negative minimum value
sns.scatterplot(x=[min_index],
            y=[minvalue],
            color='red',
            s=100,
            edgecolor='lightgrey',
            linewidth=1.5,
            label=f'Negative Minimum: {minvalue}')

# Plotting the new minimum value
sns.scatterplot(x=[min_index],
            y=[newmin],
            color='orange',
            s=100,
            edgecolor='lightgrey',
            linewidth=1.5,
            label=f'New Minimum: {newmin}')

 # Plotting horizontal lines at the negative and new minimum values
plt.axhline(y=minvalue,
         color='red',
         linestyle='--',
         label='Boundary')


plt.axhline(y=newmin,
         color='red',
         linestyle='--')


 # Set the limit close to the values
plt.ylim(-20.5, 2)
plt.xlim(-500, 11000)


plt.title('Decision Function Minimum', fontsize=14)
plt.xlabel('Index', fontsize=14)
plt.ylabel('Decision Function Value', fontsize=14)


plt.grid(True)
plt.legend()


plt.show()
```

# Boundary limits

To counteract for the multiplication of the testing set with the scalar *x*, the boundary limits are also multplied by *x*, ensuring accuracy.

```
bmin = float(newmin * x)
bmax = float(maxvalue * x)
print(f'The new minimum is', {bmin})
print(f'Therefore, the range that defines human web browsing activity (after multiplication with x) is:', [bmin, bmax])
```

## Comparing distances

To determine if the new input data defines human browsing activity, it is compared to the training data. The new input is denoted as the testing data. The comparison is done using a distance metric, this project explores Euclidean distance.

This section explores:

- <u> The transposition of the training and testing data:</u>

Both sets are transposed as the focus is comparing the features. The distance metrics compare over rows hence, by transposing the sets, the similarity metrics are comparing the features.

- <u> The calculation of the euclidean distance:</u>

The *cdist* function is used to compare the distances between all pairs of rows from the two sets. This is in reference to [5].

- <u> Whether the euclidean distances are within the boundary range:</u>

A new variable *inrange*, is a boolean array [6] that checks whether the euclidean distances are within the boundary range. This means that the distances are greater or equal than the minimum and smaller or equal than the maximum.

- <u> The final classification of the testing data:</u>

The distances within the range are counted in and labelled as 0 and 1 as per the requirements of the project. The counts are then compared to determine the majority, meaning that if there are more "True" distances than "False" ones, the testing data, hence, the client (as per assumption 1 in the report) is classified as 1 or 0.

```
# The transpose of the training data
B_trtranspose = B_train_scaled.T


# The transpose of the testing data
B_tetranspose = B_test_scaled.T
# Calculating the Euclidean distance
y_eucl = cdist(B_trtranspose, B_tetranspose, metric='euclidean')


print("The Euclidean distance of the two sets is:\n", y_eucl)
# Printing the number of values in the DataFrame
print(f"Number of values in y_eucl: {len(y_eucl)}")


# Checking if the distances are larger than the range's minimum and smaller than the
maximum
inrange = (y_eucl >= (bmin)) & (y_eucl <= (bmax))


# View if they are within range or not
print('Checking whether distances are within the range:', inrange)
```

```
# Summing up the all the distances in the range
onecount = np.sum(inrange)


# Summing up all the distances outside of the range using bitwise NOT operator '~'
zerocount = np.sum(~inrange)


# Calculating the majority
if onecount > zerocount:
    majority = "1"
else:
    majority = "0"


print(f"Number of 1s: {onecount}")
print(f"Number of 0s: {zerocount}")
print(f"The model determines that {majority} is the majority.")
```

## Conclusion

Therefore, as per the assumption, the web client with the testing data is labelled as a human.

To summarise, the OC SVM is trained on a dataset pertaining to human users and subsequently tested on the same dataset multiplied by a scalar *x*, intended to represent the privacy-preserving attribute of the ZKML framework. Using the training data in the decision function, the minimum and maximum limits are established, which are then adjusted by multiplied by *x* to account for the inflation in the testing set. This process defines the boundaries-range of the model. The features of both sets, portrayed as rows by the method of transposition, are compared using the Euclidean distance, and assessed as to whether they fall within the range defined previously. The majority count then classifies whether the web client is a human or not. Due to the dataset containing one class that defines human behaviour, the model never takes bot data as an input and therefore cannot definitively classify the client as a bot.

## References

[1] Sanchez, P. M. S. (2020) BEHACOM, Mendeley Data. Mendeley Data. Available at: https://data.mendeley.com/datasets/cg4br62535/2 (Accessed: 26 September 2024).

[2] scikit-learn.org (2019). MinMaxScaler. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html [Accessed: 26 September 2024]

[3] scikit-learn.org. (2024). sklearn.svm.OneClassSVM — scikit-learn 0.24.2 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html [Accessed 26 Sep. 2024].

[4] Pydata.org. (2024). seaborn.scatterplot — seaborn 0.13.2 documentation. [online] Available at: https://seaborn.pydata.org/generated/seaborn.scatterplot.html [Accessed 27 Sep. 2024].

[5] Scipy.org. (2024). cdist — SciPy v1.14.1 Manual. [online] Available at: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cdist.html [Accessed 27 Sep. 2024].

[6] Buffalo.edu. (2017). Boolean numpy arrays — MTH 337. [online] Available at: https://www.math.buffalo.edu/~badzioch/MTH337/PT/PT-boolean_numpy_arrays/PT-boolean_numpy_arrays.html#:~:text=A%20boolean%20array%20is%20a,numpy%20as%20np%20a%20%3D%20np. [Accessed 27 Sep. 2024].

# Appendix C

The code used for users 0, 2, 3, 5, 6, and 8 is the same with the only difference of the dataset being loaded. The following lines of code are for user 2. To run this for other users, the only thing that needs to be adjusted is loading the correct dataset. These files are also in the submission of the project on Moodle.

---

```
# Loading all necessary libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.svm import OneClassSVM

from sklearn.preprocessing import MinMaxScaler

# Loading the dataset of user 2

User2 = pd.read_csv('User2_BEHACOM.csv', encoding='latin1')

# Dropping the unnecessary columns

B = User2.drop(columns=['current_app', 'penultimate_app','USER', 'timestamp'])

# Defining training set

B_train = B


# Random scalar

x = 50


# Defining the testing data

B_test = B * x

# Scaling the sets

scaler = MinMaxScaler()
```

```
# The training data is fitted and transformed
B_train_scaled = scaler.fit_transform(B_train)


# The testing data is transformed
B_test_scaled = scaler.transform(B_test)


print(B_test_scaled)
print(B_train_scaled)
# Constructing the One Class SVM model
model = OneClassSVM(kernel='rbf', gamma='scale', nu=0.1)
# Fitting the model with the training data
model.fit(B_train_scaled)
# Deriving the decision function values
decisionf = model.decision_function(B_train_scaled)
sns.set(style='whitegrid', palette='muted')


plt.figure(figsize=(10, 6))


# Plotting the decision function output data
sns.scatterplot(x=range(len(decisionf)),
                y=decisionf,
                color='purple',
                marker='o',
                s=30,
                label='Decision Function Values',
                alpha=0.6)
```

```
# Customising the plots

plt.title('User 2', fontsize=14)

plt.xlabel('Index', fontsize=14)

plt.ylabel('Decision Function Value', fontsize=14)

plt.grid(True)

plt.legend()

plt.show()
```

# Appendix D

The following are the meetings of the author with the supervisor throughout the project.

Meeting 1: 3 July 2024

**What I've done:**

- Literature review -found relevant research
- Examples of other ZKML applications - vCNN, ZEN, zkCNN, ZENO, ZKML(as in proposal)
- Suitability small analysis of ZKP with captcha and ML with captcha

Notes:

- Just because there is a lot of research on zkSNARKS doesn't mean that should be the one i choose - it makes sense there is a lot of research, its an older topic
- zkSNARKS require trusted setup - only bulletproof doesn't require a trusted setup
- For v3: 0.9 says that the user is human and in case of a ddos then if you start falling from 0.9 then you become bot.
- Need to think about details that will define the parameters in ZKP but also the algorithms in ML.
- Look at captcha v3 gives a score to the user that entered the page -this is details.

Next steps:

- Look at specification requirements we set for v4 based also on v3
- Which zero knowledge could we use?
- From this zkp which ml algorithm could we use?

**To do:**

- Look at v3 attacks, What characteristics about v3 can you find from there?
- Design v4: define specification requirements based on v3 as well

Specification requirements for v4:

- Design model so that when someone goes in it can identify if someone is bot or human
- Needs to be an interaction - this is how zkp will get verification

- Maybe v4 doesnt say the score
- Assume the user learns the score - how can you use this

Question:

- 1. “Zero knowledge for Machine Proof” by Zhang paper cannot be obtained.
- keynote talk, not something I can use, can take information from his other papers like decision trees from relevant references.

---

Meeting 2: 10 July 2024
What I have done:

- Have written how Bulletproofs and zkSNARKS work (zkp.doc)
- Have seen how reCAPTCHA works (specification requirements.doc)
- Have attempted to describe how v4 would work but have issues.

Described two scenarios 1st with zkSNARKS and 2nd with Bulletproofs.

- Overall I feel like I need another week to understand the information in a better way.

Notes:

- Meeting notes are crucial in the dissertation report, need to take notes every meeting or penalty.

On work i’ve done:

- Scenarios so far are good but are of a higher level than needed.
- Need to take a step back - add mathematics to give more value to final report
- How do you define the circuit mathematically

What I need to do:

- Read up on all understand
- Don’t think about connecting with http etc because it’s too much focus on the start
- Start defining every step numerically in simple terms and extend/advance it later.
- Check about parameters
- Only first parts

---

Meeting 3: 17 Jul 2024

What I've done:

- Did setup and commitments for scenarios for zksnarks, bulletproofs

Notes:

- Response token specification document you can follow notations and preliminaries the same way you can response token needs to be r not rt because there's another token difference with range R.
- Pedersen commitment doesn't matter for range can use anything really since it is integer so I can use any range like 1 to 10
- For module R (real numbers) need to be positive
- Need a reference still from bulletproofs polynomial to cross reference to show integrity
- Look at what the rest have to understand b-x
- Find example of a zk snark in articles
- Zk snark need more detailed on circuits

What I need to do:

- polynomial construction more details are needed for various polynomials
- Zksnarks need more details on the circuits in - deeply explore
- Practical applications to do what is needed

---

Meeting 4: 24th July 2024

What i've done:

- How to develop polynomial for zkSNARK
- How to develop circuits
- What they did on past ZKML
- Scenario for captcha

Q:

How do i express computation needed for zkp when the y output is produced after going through the ML model → lack of information here.

Notes:

- Graph is good
- Recall or sensitivity is how related what you have is to the data set
- Precision is the precision accuracy from formula there you see percentage of accuracy
- What does accuracy mean, does it mean that he is acting like human
- How can this be setup
- Depends on what y is supposed to be - based on this, y could be sensitivity or recall
- Could be percentage 50-60%
- Define accuracy

What to do:

- Explore scenario more - what do we consider as the secret
- Continue from meeting 3 notes - derive polynomial
- Think about y output
- Precision or recall/sensitivity

---

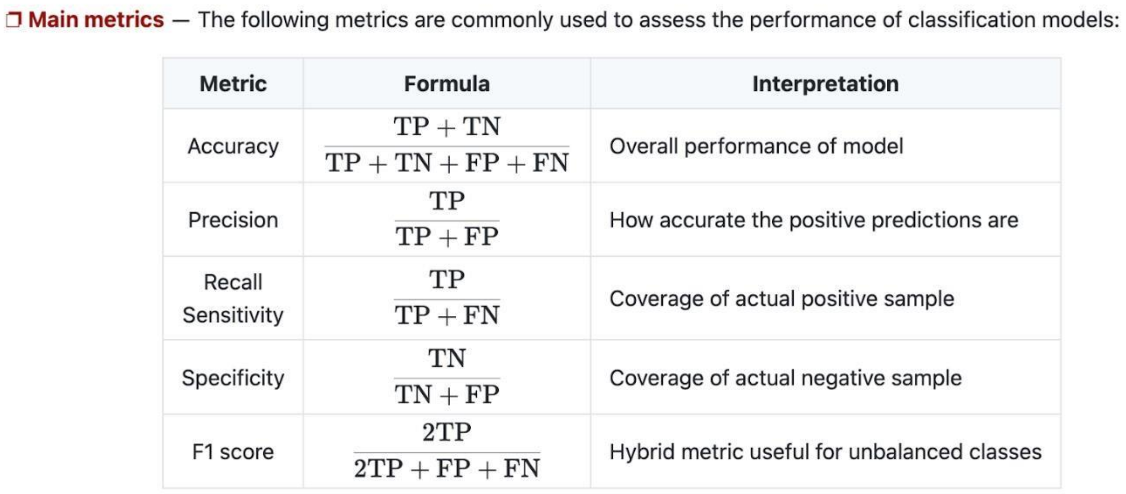
Meeting 5: 7 Aug 2024

What I have done:

- Explored what output y is
- Looked into accuracy, precision, sensitivity

Notes discussed:

- Potential use of metrics like Euclidean distance or z-score.



❐ **Main metrics** — The following metrics are commonly used to assess the performance of classification models:

| Metric | Formula | Interpretation |
|---|---|---|
| Accuracy | $\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$ | Overall performance of model |
| Precision | $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$ | How accurate the positive predictions are |
| Recall Sensitivity | $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$ | Coverage of actual positive sample |
| Specificity | $\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$ | Coverage of actual negative sample |
| F1 score | $\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$ | Hybrid metric useful for unbalanced classes |

- The ML can take y and see if it is classified(as human) or misclassified(as bot) - linear regression, answer doesn't have to be from 0 to 1.
- [General Model]: With linear regression, if you have the same number of instances in the training model with the testing model you can see if the ML classifies the client as benign normal or malicious-bot. Create the same instances with testing and see if they fall from left or right. Each side is bot or human, from this can see majority vote. This way there won't even be the need for a score.
- The secret is client's behavioural data. Changing domain technique: Domain of 0-100 can be changed to 1000-2000. → 1300 is very different to 1400 compared to 30 with 40.
- Why behavioural data needs to be secret:
- Attack scenario if someone intercepts my behavioural data and uses it on their machine can then have their bot be classified as human
- Privacy preservation of client data

[General Model]:

1. Server gives client value x
2. Client uses x WITH the behavioural data (**B** vector) to produce y.
3. Client sends y to server
4. Server receives y, extracts the part of the behavioural data.

Behavioural data is the common secret, no one sends the actual data to each other.

Result is developed by either:

1. Calculating y and seeing where it stands within the range that defines human.

2. Putting y through the training model and see how it is classified, is it class 1 or 0 so → malicious (bot) or benign (human)?

Server sends x as a challenge → **X IS CHALLENGE** client has to do computations with beh. Data **B** vector and send y → **Y IS RESPONSE,** so server does something with y - puts it through ML model sees classification and sends back response whether the clint is accepted into the website or not.

Implementation can be done with:

- SageMath - for big numbers - there is the server option of sageserver SageServer - Sagemath Wiki
- Need socket programming - echo server and echo client Sage Cell Server (sagemath.org)
- To check zkp ProVerif (inria.fr) - potentially

What I need to do:

- Need metrics such as Euclidean distance, z score

- Look at Kotaro's thesis about most suitable ML mentioned

- Describe, compare and investigate potential attacks for 2 methodologies discussed:

1. Client sends x to server and then calculation to find y.

2. Server gives a challenge, the client answers/calculates something and then result is produced - how ZKP works normally

Also consider different zkp protocols that can be used, zksnarks, bulletproofs

Need to define what is needed - accuracy, precision, similarity or confusion matrix

- Construct circuit **mathematically**
- Compare overhead - memory and computation between zksnark and bulletproofs.
- From Thesis-2.pdf read only Appendix B page 62 - discusses proverif and how to do it - formal way of checking that there is mutual identification, there is forward secrecy, there are no leaks.

Questions:

1. But client is not trusted so there can't be a challenge → There needs to be a witness for the cient-server connection to be established otherwise unsuccessful cannot avoid
2. Should I continue and find which ML algorithm is most suitable? → does not change anything but what we need is the comparison of y whether its in a range or not this could be a mixture of many metrics.

---

Meeting 6: 28/08/2024

What I have done:

- Constructed the high level scenario with v4 captcha as i did in the first meeting.

Discussion:

- Comment: SageMath is taking too long when it comes to calculations with the dataset → shorten the dataset/ dont take all of it or put loop i for 5 - dont let the entire thing run of course it will take time. With decimals be careful when comparing because this could go on for too long.
- accuracy in my code not correct, I'm looking for similarity for the one class that exists and check how similar this unseen data is to the 1 class. Need to see what min

is, max and average to get the human threshold threshold = [min,max] see what the similarity score is for the new unseen data.

Next steps:

- Since there is only one label on the dataset:
- Try to find dataset with malicious bot activity on the web
- Use this paper as a reference  - uses 1 class SVM and 1 class NN

Questions:

- When developing the code should I use KNN ready code or the euclidean distance then calculate steps manually- step by step?
- The dataset's last column denotes the user so user0, user 1,2 etc this is confusing me → make a copy and delete this column not needed.
- The dataset only has 1 label denoting human/ no malicious data but MLs need at least two, what happens now → ideally need from bot too, spend time searching for a bot dataset but becomes complicated with the features → this is a limitation

---

Meeting 7: 11th September 2024

Discussion:

Regarding the code:

- I have split the training and testing data as discussed in the previous meeting
- Have yet to do the decision function
- Min comes out as always negative no matter the hyperparameter tuning - so the decision function must take in consideration the outlier as well.

Proposed fixing:

- need to see what the value before this min outlier is - take zscore to see what the median between the outlier and its previous value is.
- Issue might be with the predict line
- Check one class paper shared.

- Issue with scaling the testing part - when multiplied by c it seems like the scaling makes the testing the same data as the training, clearly the testing data is being descaled
- Take assumptions.

To do:

- Graph the test in model and find the outlier (minimum value) then try to see the average - zscore of euclidean between this outlier and the value previously hmm
- Zscore will give average of how similar these things are meaning when i see the outlier i need to see the possibilities of this outlier for the previous to not be an outlier

---

Meeting 8: 18th September 2024

What I've done:

- Could not find bot dataset
- Still working on the code

Notes:

Comments on code:

- The problem is most likely with test and train sets.
- Should: 1. Train model, 2 test model 3. use similarity
- Problem is with the variables, becomes confusing, print them to see
- Have to train and test model with same user
- Number have to be scaled within a range → research this

Question: How to derive the score since I have many values → use their mean

Next meeting is the last meeting.

---

Meeting 9: 11th September 2024

Discussion:

Regarding the code:

- I have split the training and testing data as discussed in the previous meeting
- Have yet to do the decision function
- Min comes out as always negative no matter the hyperparameter tuning - so the decision function must take in consideration the outlier as well.

Proposed fixing: need to see what the value before this min outlier is - take zscore to see what is the median between the outlier and its previous value

Issue might be with the predict line

Check one class paper shared.

- Issue with scaling the testing part - when multiplied by c it seems like the scaling makes the testing the same data as the training, clearly the testing data is being descaled.
- Take assumptions.

Question:

- Can I use the decision function to get the outliers? → yes but might have to do the decision instance for each row in a for loop method
- One of the objectives says to evaluate the effectiveness of the model, will this be done with metrics? → since it is for proof of concept explore complexity, tradeoffs of security between efficiency - and security. We prioritised the security of the client's data, can compare this work if it was done in other studies. What other metrics did the other studies take? Metric 1 can be memory, 2 processing power, 3 runtime etc check, security evaluation
- Can i not use similarity and use svm as is to just evaluate if its outlier or not → yes but svm classifies the new data as benign or not benign, since this is one class svm it

cannot with certainty classify that the user is bot since there is no corresponding bot data.

- For classification in ZKP, we have setup/commitments, proof algorithm and verification is ML.
- One objective is that i have to examine which zkp is suitable, how do i justify this → have to look back at notes, its good to backup your decisions with scientific articles “based on studies”, limitations etc. its not suitable. So 1. Look at notes and 2. Think if i can backup

To do:

- Graph the test in model and find the outlier (minimum value) then try to see the average - zscore of euclidean between this outlier and the value previously.