

APRIL 2018

---

# Location Privacy Tool - A Web Demo

---

LAUREN HELEN TAPLIN

114377971

FINAL YEAR PROJECT - BSc IN COMPUTER SCIENCE



Mentor: Paolo Palmieri

University College Cork  
Department of Computer Science

## **Abstract**

Location privacy is a growing issue in cyber security. The Spatial Bloom Filter is one of the many privacy-preserving protocols aimed at mitigating this threat to personal privacy. The goal of this project is to develop a graphical web demo that would explain to non-experts what a Spatial Bloom Filter is and how it works.

The Spatial Bloom Filter is a data structure based on bloom filters, designed to store location information. The protocol keeps the user's exact location hidden but allows the provider of the service to learn when the user is within predefined specific points of interest. At the same time, the points and areas of interest remain oblivious to the user.

This project uses Flask to develop a simple and intuitive web demo to display the Spatial Bloom Filter data structure, and the cryptographical protocols used, explaining how the SBF protocol works.

## Declaration of Originality

In signing this declaration, you are conforming, in writing, that the submitted work is entirely your own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award.

I hereby declare that:

- this is all my own work, unless clearly indicated otherwise, with full and proper accreditation;
- with respect to my own work: none of it has been submitted at any educational institution contributing in any way to an educational award;
- with respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

---

Lauren Taplin

---

Date

## **Acknowledgement**

Thank you to Paolo Palmieri providing the opportunity for work on this project. His help, guidance and advice were greatly appreciated. To friends and family for their support and input on the design of the web demonstration and acting as non-expert test subjects.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	Goal of the Project . . . . .	1
1.3	Technologies . . . . .	3
<b>2</b>	<b>Background</b>	<b>4</b>
2.1	How can location data be used? . . . . .	4
2.2	Literature Review . . . . .	5
2.2.1	Article 1 . . . . .	5
2.2.2	Article 2 . . . . .	6
<b>3</b>	<b>Analysis</b>	<b>7</b>
3.1	Objectives of the Project . . . . .	7
3.2	Cork Dataset . . . . .	8
3.3	Web Demonstrations . . . . .	9
<b>4</b>	<b>Design</b>	<b>11</b>
4.1	Technologies . . . . .	11
4.1.1	Python3 . . . . .	11
4.1.1.1	ast . . . . .	12
4.1.1.2	base64 . . . . .	12
4.1.1.3	csv . . . . .	13
4.1.1.4	hashlib . . . . .	13
4.1.1.5	numpy . . . . .	14

4.1.1.6	pathlib . . . . .	14
4.1.1.7	random . . . . .	15
4.1.1.8	sys . . . . .	15
4.1.2	Flask . . . . .	16
4.1.3	MaterializeCSS . . . . .	16
4.2	Tools . . . . .	16
4.2.1	Version Control . . . . .	16
4.2.2	GitHub . . . . .	17
4.2.3	PyCharm . . . . .	17
4.3	Server - AWS . . . . .	18
<b>5</b>	<b>Implementation</b>	<b>19</b>
5.1	Back-end . . . . .	19
5.1.1	Overview . . . . .	19
5.1.2	Insertion . . . . .	19
5.1.3	Statistics . . . . .	20
5.1.3.1	The Spatial Bloom Filter . . . . .	20
5.1.3.2	Areas of Interest . . . . .	21
5.1.4	Layouts and Templates . . . . .	22
5.1.4.1	Flask . . . . .	22
5.1.4.2	Display . . . . .	23
5.1.4.3	Test Data . . . . .	24
5.1.4.4	Check . . . . .	26
5.2	Front-end . . . . .	27
5.3	Server . . . . .	29
5.3.1	AWS Setup . . . . .	29
5.3.2	Apache . . . . .	30
5.3.3	Difference Between AWS and Local . . . . .	32

<b>6</b>	<b>Evaluation</b>	<b>33</b>
6.1	Testing . . . . .	33
6.1.1	Unit Testing . . . . .	33
6.1.2	Exploratory Testing . . . . .	33
6.1.3	Known Errors . . . . .	34
6.2	User Friendly . . . . .	34
6.3	Graphs . . . . .	36
6.3.1	False Positive Probability . . . . .	36
6.3.2	Sparsity and Hash Collisions . . . . .	36
6.4	Tables . . . . .	37
6.4.1	The Spatial Bloom Filter . . . . .	37
6.4.1.1	False Positive Probability . . . . .	37
6.4.1.2	Sparsity . . . . .	37
6.4.1.3	Hash Collisions . . . . .	37
6.4.1.4	Hash Family . . . . .	37
6.4.2	Areas of Interest . . . . .	38
6.4.2.1	Members . . . . .	38
6.4.2.2	Cells Used . . . . .	38
6.4.2.3	Potential Cells . . . . .	38
6.4.2.4	Self Collision . . . . .	38
6.4.2.5	Emersion . . . . .	38
6.4.2.6	False Positive Probability . . . . .	39
6.4.2.7	Inter Set Error Probability . . . . .	39
<b>7</b>	<b>Conclusions</b>	<b>40</b>
7.1	Summary and Accomplishments . . . . .	40
7.2	Future Developments . . . . .	41
7.2.1	Edit Details . . . . .	41

7.2.2	Test Data . . . . .	41
<b>A</b>	<b>Appendix</b>	<b>42</b>
A.1	False Positive Probability Graph . . . . .	42
A.2	Sparsity Graph . . . . .	43
A.3	Hash Collision Graph . . . . .	44
	<b>References</b>	<b>44</b>



# 1 Introduction

## 1.1 OVERVIEW

Privacy is a growing issue in today's world. With more incidents regarding the misuse of user's data coming to light, more actions are being taken to protect users personal data. The focus of this project is the protection of users location privacy.

Global Positioning System's (GPS) enabled technologies such as smart-phones, cars, cameras, etc. allow the monitoring of a users location placement to be more precise. As devices get smaller and more affordable, the tracking of users is becoming more accessible. Social networks such as Facebook, use the location data to target specific advertisements to their users [1] whereas services such as Google, can confirm if their users attended events marked on their Google Calendar by tracking their user's location through Google Maps. [2]

The Spatial Bloom Filter is a purposed protocol that could be used to prevent this round-the-clock surveillance of users, preserving their location privacy.

## 1.2 GOAL OF THE PROJECT

The goal of this project is to design a web demonstration, with the purpose of explaining to non-experts what the Spatial Bloom Filter is and how it works.

The web demonstration accomplishes this with a simple graphical web interface using helpful tools such as MaterializeCSS and Flask to help present the Spatial Bloom Filter protocol in a clear and concise manner that is also user-friendly. Test data containing the coordinates of area's of interest is provided to populate the

Spatial Bloom Filter which allows it to be queried and return meaningful results along with explanations of those results.

Overall, the web demonstration would allow non-experts to interact with the Spatial Bloom Filter and gain an understanding of how the protocol works.

### 1.3 TECHNOLOGIES

[illegible]

**Figure 1.1:** Web Demo.

Over the course of the project, technologies such as Flask and MaterializeCSS were used to achieve its goals. Implementing the Spatial Bloom Filter and using front and back-end web frameworks to design a user-friendly graphical web display.

Figure 1.1 shows the demo of the Spatial Bloom Filter.

## 2 Background

As mobile phones become smaller, cheaper and smarter, privacy is becoming a more significant issue. Several technologies can be combined to determine the location of mobile devices: use of the mobile phone's signal, GPS, and Wi-Fi [3]. This combination allows location services to become more precise.

### 2.1 HOW CAN LOCATION DATA BE USED?

By tracking location data, it makes it possible to inquire about an individuals movements that they do not wish to be public knowledge [4].

- Did they visit an abortion clinic?
- Did they see an AIDS counsellor?
- Which church do they attend? Which gay bar?
- Where did they go at lunchtime? With whom?
- Who did their ex-partner go to dinner with last week?

The gathering of this information is collected silently and cheaply through applications which can then analyse the data.

## 2.2 LITERATURE REVIEW

### 2.2.1 Article 1

**Article Name:** Location privacy without mutual trust: the Spatial Bloom Filter [5]

**Background:** This paper is the subject of this project. The idea of this paper is the creation of a data structure which is based on bloom filters to store location information. The purposed protocol would keep the user's exact location private; however, the service provider can learn when the user is in a predefined area of interest. Meanwhile, these area's of interest remain unknown to the user.

**Conclusion:**

- The modified bloom filter is suited to location information as long as it is a set-based format.
- Area's of interest values are encoded when inserted into the Spatial Bloom Filter which allows sensitive areas such as military bases hidden from their users.
- The user's exact position is never known to the provider of the service. They only identify when said user is within one of their predefined areas.

### 2.2.2 Article 2

**Article Name:** Probabilistic properties of the spatial bloom filters and their relevance to cryptographic protocols [6]

**Background:** This paper is built upon the previous paper "*Location privacy without mutual trust: the Spatial Bloom Filter*". It examines the Spatial Bloom Filter probabilistic properties. New metrics are defined, for instance, emersion. These metrics are beneficial for achieving Spatial Bloom Filter characteristics such as the inter-set error rate.

**Conclusion:**

- New metrics provide a better understanding of the Spatial Bloom Filter characteristics.
- Emersion and inter-set error probability can be implemented to provide more information about the Areas of Interest.

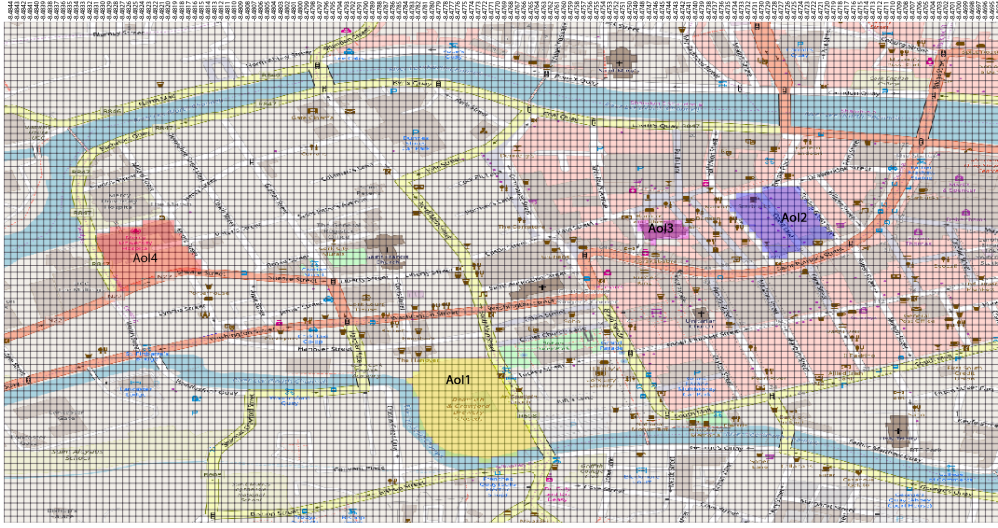
## 3 Analysis

### 3.1 OBJECTIVES OF THE PROJECT

The project aims to:

- Present a web demonstration of a Spatial Bloom Filter.
- The demonstration would allow users to import test data based on the city of Cork and insert it into the filter.
- Display the Spatial Bloom Filter, and its contents to allow users to visualise how the data is represented once inserted.
- Provide the functionality of querying the Spatial Bloom Filter with values such as the user's current position.
- Show the results of the search query and highlight their corresponding values in the Spatial Bloom Filter.
- Explain the results and which state which area of interest they belong to if any.
- Provide a feature which allows users to edit the Spatial Bloom Filter details by changing the hash functions used.
- Provide the statistics of the Spatial Bloom Filter such as the false positive probability rate to showcase the accuracy of the results.
- Provide the statistics of the areas of interest such as self-collisions and its emersion to display which area is more accurate.

## 3.2 CORK DATASET



**Figure 3.1:** Map of Cork with Areas of Interest

To ensure that the Spatial Bloom Filter produced a meaningful demonstration, creating a test data set was necessary.

After selecting a region to use as an example, there were four areas of interest extracted from the map. These areas varied in size to produce different statistics for each Area of Interest once inserted into the Spatial Bloom Filter.

Figure 1.1 shows the Map of Cork City with four Areas of Interest highlighted.

The coordinates obtained from the areas of interest had to be accurate to 0.0001 of a degree to improve the accuracy of the results. Each of the four areas had a set of coordinates associated with them recorded in a CSV file.

The Spatial Bloom Filter would read in the coordinates from a CSV file. The separator used for the CSV file is a comma, so each value is read in as 1, 51.8954#-8.4772. The coordinate was hashed to produce an index and then insert the area of interest value at that index.



### 3.3 WEB DEMONSTRATIONS

The base data structure of a Bloom filter is a **Bit Vector**. Here's a small one we'll use to demonstrate:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----

Each empty cell in that table represents a bit, and the number below it its index. To add an element to the Bloom filter, we simply hash it a few times and set the bits in the bit vector at the index of those hashes to 1.

It's easier to see what that means than explain it, so enter some strings and see how the bit vector changes. Fnv and Murmur are two simple hash functions:

Enter a string:

fnv:  
murmur:  
Your set: []

**Figure 3.2:** Insert into Bloom Filter.

Test an element for membership:

fnv:  
murmur:

Is the element in the set? no

Probability of a false positive: 0%

**Figure 3.3:** Query the Bloom Filter.

Reviewing similar tutorial demonstrations of bloom filters contributed to the approach taken to designing the graphical interface of the spatial bloom filter.

By studying tutorials such as Bloom Filters by Example [7], see figure 3.4 below, which included an insert function and a check function, see figure 3.2 and figure 3.3 above, the implementation of the demonstration for the Spatial Bloom Filter similar.

The Spatial Bloom Filter demonstration includes:

- an import function,
- a check function,
- statistics about the filter itself and;
- statistics about the Areas of Interest.

However, the original implementation of the filter created with a maximum size of 64 cells. Though with a total of 458 coordinates and three hash functions, the false positive probability rate was too high to provide a meaning explanation of how the Spatial Bloom Filter worked.

[简体中文](#)

## Bloom Filters by Example

A Bloom filter is a data structure designed to tell you, rapidly and memory-efficiently, whether an element is present in a set.

The price paid for this efficiency is that a Bloom filter is a **probabilistic data structure**: it tells us that the element either *definitely is not* in the set or *may be* in the set.

The base data structure of a Bloom filter is a **Bit Vector**. Here's a small one we'll use to demonstrate:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----

Each empty cell in that table represents a bit, and the number below it its index. To add an element to the Bloom filter, we simply hash it a few times and set the bits in the bit vector at the index of those hashes to 1.

It's easier to see what that means than explain it, so enter some strings and see how the bit vector changes. Fnv and Murmur are two simple hash functions:

Enter a string:

fnv:  
murmur:

Your set: []

When you add a string, you can see that the bits at the index given by the hashes are set to 1. I've used the color green to show the newly added ones, but any colored cell is simply a 1.

To test for membership, you simply hash the string with the same hash functions, then see if those values are set in the bit vector. If they aren't, you know that the element isn't in the set. If they are, you only know that it *might* be, because another element or some combination of other elements could have set the same bits. Again, let's demonstrate:

Test an element for membership:

fnv:  
murmur:

Is the element in the set? no

Probability of a false positive: 0%

Figure 3.4: Bloom Filter by Example website layout

## 4 Design

This contains a list of the technologies and tools that were used throughout the project.

### 4.1 TECHNOLOGIES

#### 4.1.1 Python3

Python is a simple but powerful programming language [8]. Its use of efficient data structures and hash function made it the ideal programming language to use to implement the Spatial Bloom Filter data structure.

Not only is it a useful language for scripting but also for quick application deployment. By using native libraries and third-party frameworks, creating a website that could interact with the Spatial Bloom Filter implementation and display it on screen was ideal. This functionality made Python the primary language of this project.

#### 4.1.1.1 ast

ast is a Python package that helps applications to process abstract syntax grammar. In the context of this project, it was used to convert a string representation of a list to a list. See example; listing 1.

```
>>> import ast
>>> lst = '[1, 2, 3, 4]'
>>> lst = ast.literal_eval(lst)
>>> lst
[1, 2, 3, 4]
```

**Listing 1:** ast example

#### 4.1.1.2 base64

base64 is a Python package that provides functions to encode binary data to ASCII characters and decode it back. In the context of this project, it was used to decode a Base64 encoded hash salt into bytes. See example; listing 2.

```
>>> import base64
>>> salt_file = 'hash_salt/hash_salt'
>>> hash_salts = []
>>> salt_file_lines = salt_file.readlines()
>>> for line in salt_file_lines:
...     hash_salts.append(base64.b64decode(line.strip()))
```

**Listing 2:** base64 example

#### 4.1.1.3 csv

csv is a Python package that helps applications to read and write data in CSV format. In the context of this project, it was used to read in the coordinates of the areas of interest which were stored in a comma separated CSV file. See example; listing 3.

```
>>> import csv
>>> with open('dataset/cork.csv', 'r') as dataset_file:
...     dataset_reader = csv.reader(dataset_file, delimiter=',')
```

**Listing 3:** csv example

#### 4.1.1.4 hashlib

hashlib is a Python package that implements secure hash digest algorithms. In the context of this project, it was used to hash the coordinate before insertion into the file. See example; listing 4.

```
>>> import hashlib
>>> buffer = bytes(''.join([chr(ord(a) ^ b) for (a, b) in zip(
                                element, hash_salts)]), 'latin-1'
                    )
>>> m = hashlib.sha256()
>>> m.update(buffer)
>>> m.digest()
```

**Listing 4:** hashlib example

#### 4.1.1.5 numpy

numpy is a Python package used to represent multidimensional array [9]. It is more compact than a Python list thus saving space. In the context of this project, numpy was used to represent the Spatial Bloom Filter as the size of the array could be considerable large, for example,  $2^{32}$ . See example; listing 5.

```
>>> from numpy as np
>>> filter = np.array(np.repeat(0, num_cells), dtype=np.uint8)
```

**Listing 5:** numpy example

#### 4.1.1.6 pathlib

pathlib is a Python package used to present file system paths in line with the systems operating system. The module Path was used to determine if the file existed. See example; listing 6.

```
>>> from pathlib import Path
>>> aws = "/var/www/demo/hash_salt/hash_salt"
>>> if Path(aws).is_file():
...     return aws
>>> return "hash_salt/hash_salt"
```

**Listing 6:** pathlib example

#### 4.1.1.7 random

random is a Python package use to generate pseudo-random numbers. The module randint was to generate random natural integers. In the context of this module, it was used to select random coordinates to test if they produced incorrect values when the Spatial Bloom Filter was queried so they could be used as test data. See example; listing 7.

```
>>> from random import randint
>>> randint(1, 100)
67
```

**Listing 7:** random example

#### 4.1.1.8 sys

sys is a Python package that provides information about the Python interpreter. The module byteorder is used to indicate the native byte order of the platform. In the context of this project, it was used to determine the byte order used to represent an array of bytes as an integer. See example; listing 8.

```
>>> from sys import byteorder
>>> int.from_bytes(b'\x00\x10', byteorder=byteorder)
4096
```

**Listing 8:** sys example

### **4.1.2 Flask**

Flask is a Python microframework used for deploying web applications [10]. Its use of simple boilerplate code means it is ideal for getting an application up and running. Flask is relatively minimal and bare-bones and allows the developer to add-ons any extras themselves.

For a simple web demonstration, Flask was the preferred option over Django which provides the developer with an admin panel and with a database interface which was unnecessary for the project.

### **4.1.3 MaterializeCSS**

Materialize is a front-end framework designed by Google based on Material Design [11]. It is a CSS framework that makes styling web pages much more manageable.

By stating the MaterializeCSS classes in the HTML tags, the CSS is applied to the web page but can be overwritten in the CSS file. This simple framework constructs a creates a functional and modern web design fast and efficiently.

Its responsive design allows the web page to redesign itself depending on the devices screen size making it fully compatible across devices such as PCs, tablets and mobile devices. It is also cross-browser compatible.

## **4.2 TOOLS**

### **4.2.1 Version Control**

Git is a free and open source tool for version control [12]. It detects changes made to file on the local file system and documents those changes which can be pushed to a remote repository.



The Git feature of branching allows changes to be made to the code base without affecting the master branch. Those changes will not affect the main code base until they are merged back into the master branch.

It also records every change to the code base which will allow it to be reverted to a previous state if there are any issues with the current implementation.

By using the remote repository, the project was not tied to a single computer and accessed from anywhere on any machine. It also meant that there was a backup in case any issue arose with the local machines.

### **4.2.2 GitHub**

GitHub is a free service where remote code repositories are hosted [13]. It is web service interface service for the version control Git.

By using Git to keep the repo up to date, GitHub has the code base which can be accessed from anywhere with an internet connection. GitHub is a visual representation of Git, which tracked and changes to files. GitHub can show the history of any file as well as any changes that have been made to them. It also displays the open branches and what file have been changed but have not yet been merged into the master branch.

GitHub is simple to use and an easy way to store code.

### **4.2.3 PyCharm**

PyCharm is a python programming IDE (integrated development environment) [14]. Its syntax highlighting, code completion, PEP8 standard warning and project navigation make it an ideal text editor for python projects.

PyCharm is integrated with version control, so when new files are created, it asks if the file to be added to Git. Git can be used through PyCharm; without

having to use the terminal. It can create branches, add, commit, merge and push to the remote repository as well as pull from it.

Using PyCharm streamlined production as more work could be performed from within the one application.

### **4.3 SERVER - AWS**

Amazon Web Services is a platform that can host web servers. It has a free tier which is based on usage and as the server created for this project was not expecting much traffic; it did not have to scale, so no costs occurred.

To host the project's web demonstration, an EC2 instance had to be created. Each component such as the server's operating system was chosen on what was required for the project. The EC2 stances could be spun up or stopped at any moment and could be hosted on any of the available data centres.

AWS is well documented and offers many tutorials on how to use their different services. It is an excellent way to learn about servers.

## 5 Implementation

### 5.1 BACK-END

#### 5.1.1 Overview

The Spatial Bloom Filter was created using numpy array as its usage is much smaller when compared to a Python list. As the Spatial Bloom Filter will have many cells, a numpy array was the preferred option. It would compact the size and is efficient in speed and functionality [15].

#### 5.1.2 Insertion

Stored in a CSV file was the set of coordinates belonging to each Area of Interest which was read in by the Spatial Bloom Filter. For the coordinate and its Area of interest value to be inserted into the filter, it needed to be securely hashed.

First, the coordinated was XORed with the hash salt to transform it into bytes. Next, it was hashed using the selected hash function from the approved hash family and truncated.

To truncate the resulting hash digest, the cut off point is calculated based off of the size of the filter. This value is needed to determine the index to insert the associated Area of Interest value. As the filter size for the demonstration was 1024 cells,  $2^{10}$ , this defined at what point the hash digest was to be truncated, which was after 2 bytes. See listing 9.

However, if left at 2 bytes, an index error would occur. The filters size meant that the maximum bit mapping was 10 bits and but currently set to 2 bytes, 16

```
>>> (10 + 7) // 8
2
```

**Listing 9:** Calculate at what point to truncate

bits. As 10 is not a byte aligned number of bits, the excess bits from the last process byte is shifted off from the right. See listing 10.

```
>>> x = int.from_bytes(byte[:2], byteorder=byteorder)
>>> x >>= 8 - (10 % 8)
```

**Listing 10:** shifting excess bits

Each coordinate is place into the Spatial Bloom Filter three times as there is three hash functions.

The calculation of the statistics relating to the Spatial Bloom Filter and the Areas of Interest could only be performed on the completion of the insertion of the Cork data.

### 5.1.3 Statistics

There were two types of statistics calculated, first, about the Spatial Bloom Filter itself, the other, about the Areas of Interest.

#### 5.1.3.1 The Spatial Bloom Filter

The statistics about the Spatial Bloom Filter were initially gathered as a way to make an informed decision on the size of the filter. Information along the lines of the false positive probability, the sparsity and the number of hash collisions provided enough statistics to make that decision.

However, those statistics also provided more context about the Spatial Bloom Filter and explained why it would get some coordinates wrong. The statistic results

were then included in the web demonstration.

The most common statistic associated with bloom filters is its false positive probability. It is the probability that the value tested is wrongly claimed as a member of the bloom filter. The calculation of the Spatial Bloom Filter's false positive probability is implemented in listing 11.

```
def fpp():
    c = 0

    # Count non-zero cells
    for i in range(1, num_areas_of_interest+1):
        c += num_area_cells[i]

    # Devide by the total number of cells
    p = c / num_cells

    # to the power of the number of hash functions
    return pow(p, len(hash_family))
```

**Listing 11:** Spatial Bloom Filter False Positive Probability Calculation.

#### 5.1.3.2 Areas of Interest

As the statistics for the filter itself provided informative information, therefore implementing them for each Area of Interest would also produce informative data. The statistics for Areas of Interest would differ slightly from the filters in the fact that it was possible to calculate the Area of Interest chance of being overwritten or returning the wrong area.

The emersion of an Area of Interest looks at how much the area "*emerges*" from the Spatial Bloom Filter. The more the area is overwritten, the less it "*emerges*". See listing 12

```

def area_emersion(area):
if area_members[area] == 0:
    return -1
else:
    potential_cell = (area_members[area] * len(hash_family)) -
                    area_self_collisions[area]
    return (num_area_cells[area] / (potential_cell))

```

**Listing 12:** Area of Interest Emersion Calculation.

## 5.1.4 Layouts and Templates

### 5.1.4.1 Flask

Flask works similarly to jinja2; there are variables throughout the HTML code which are substituted once the template is rendered, see example HTML code; listing 13. These variables are usually strings of HTML code generated to display the Spatial Bloom filter or its statistics.

```

<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    {{ variable_name }}
  </body>
</html>

```

**Listing 13:** Example HTML templates

However, the generated HTML code is a string, and for it to render correctly, it must be converted from a string to HTML which can be achieved by using Markup. This module is imported from the flask package. See example; list 14.

```

from flask import Flask, render_template, Markup
app = Flask(__name__)
app.secret_key = 'spacial bloom filter'

@app.route('/')
def index():
    filter = get_filter_html()
    return render_template('index.html', filter=Markup(filter))

```

**Listing 14:** Basic Flask Application.

### 5.1.4.2 Display

Figure 5.1 and figure 5.2 show the differences between the statistics once the test data has been imported.

When the web demonstration is loaded initially on screen, a filter populated with zeros is represented, see figure 5.3, on the screen as well as basic statistics about the Spatial Bloom Filter.

When the test data set is imported, new HTML is generated to display the filter that is populated with Area's of Interest, see figure 5.4, and the statistics about the filter now that it contains meaningful data.

Spatial Bloom Filter Stats
Import Cork data for more stats.
Hash Family: ['md5', 'sha1', 'sha256']
Number of Cells: 1024

**Figure 5.1:** Statistics of Spatial Bloom Filter.

Spatial Bloom Filter Stats
Number of Mapped Elements: 457
Filter Sparsity: 0.2595
Number of Hash Collisions: 623
Hash Family: ['md5', 'sha1', 'sha256']
Filter False Positive Probability: 0.3898
Number of Cells: 1024
Area Stats: <a href="#">Click Here</a>

**Figure 5.2:** Initial Statistics of Spatial Bloom Filter.





not belong in any of the Areas of Interest but claim that they do.

To find the first type of test data, the Cork dataset is read in again, but instead of inserting it into the Spatial Bloom Filter, it is used to query it. If the minimum of the potential Areas of Interest returned do not match the actual Area of Interest, then it is added to a list which is displayed on the Test Data page.

To find the other type of test data, coordinated that are not in the Spatial Bloom Filter but are still within the Cork region, are randomly selected and are used to query the filter. If the minimum of the potential Areas of Interest is not a zero, then it is added to another list which is also displayed on the Test Data page. See listing 15.

```
def find_incorrect_test_data():
    indexes = []
    while len(indexes) != 100:
        indexes.append(randint(0, len(all_coors)))
    for i in indexes:
        if len(fp_coor) == 10:
            return fp_coor
        aoi = []
        check_result = check(all_coors[i])
        for hf in hash_family:
            aoi.append(check_result[hf][1])
        if min(int(m) for m in aoi) != 0:
            fp_coor[all_coors[i]] = aoi
```

**Listing 15:** Find incorrect values for test data.

#### 5.1.4.4 Check

Check SBF Values

Q Value

CHECK ✓

Results

MD5	SHA1	SHA256

**Figure 5.5:** Check Spatial Bloom Filter Layout.

The check search and check result boxes are permanently displayed on the screen. The check result states which hash functions are being used for the Spatial Bloom Filter even though no values have been queried, figure 5.5.

Check SBF Values

Q Value

CHECK ✓

Results

MD5	SHA1	SHA256
4	4	4

51.8990#-8.4820 is in Area of Interest 4.

**Figure 5.6:** Check Results.

The results of a query return three potential Areas of Interest. The three values represent the results of each of the hash functions. As the Areas of interest are inserted into the Spatial Bloom Filter in order of smallest to highest, the lower Areas of interest could potentially be overwritten. Therefore, the lowest value of the three Areas of Interest is the area where the coordinate belongs, figure 5.6. If any of the three values return a zero, then it is not in the filter.

The results return by a query are also linked to the Spatial Bloom Filter below. By clicking on any of the three Areas of Interest, it will jump to the index of where it belongs to in the filter. The areas results are highlighted in red, and when the mouse hovers above them, a tool-tip appears with information regarding which

0	1	SHA1						2	
0	4	0	2	4	4	0	1	2	4
0	0	2	1	0	1	4	4	0	2
1	0	4	2	4	4	3	1	2	0

**Figure 5.7:** Check results highlighted in filter.

hash function returned which value, figure 5.7.

## 5.2 FRONT-END

The primary issue encounter in the front end of the web demonstration was determining the size of the Spatial Bloom Filter. Initially, the filter contained only 16 cells. However, as the Cork dataset grew, every coordinate insert overwrote another previous coordinate.

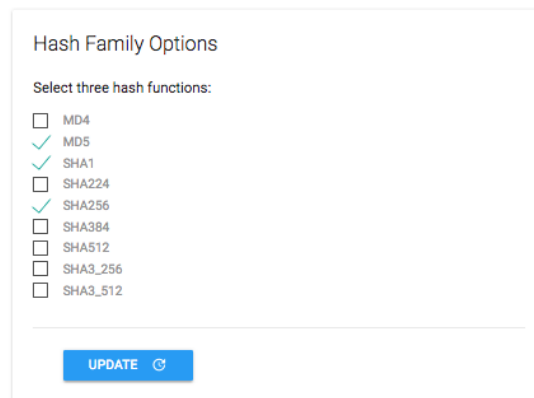
The filter increased to 64 cells which solved the issue for a time. Once the Cork dataset was finalised as well as the number of hash functions ran, 64 cells became inadequate.

The official, finalised coordinate count reached 458 coordinated which were entered into the Spatial Bloom Filter three times as a result of the three hash functions. 1,374 values were inserted in total.

The number of cells increased to 1,024. This number is still not enough cells to ensure accurate results, however, as this was a web demonstration, some test values needed to be incorrect to demonstrate what would happen in situations when the Spatial Bloom Filter was wrong.

The Spatial Bloom Filter when it was set to 1,024 cells had a false positive probability rate of 0.3898. When the filter was set to 512 cells, the false positive probability rate was 0.7785, which was too high to give any meaningful results. The same could be said when it was set to 2,048 cells; the false positive probability was 0.1164 was too low.

The next challenge was to design a filter that could be displayed on a web page that was of size 1,024 cells. When the Spatial Bloom Filter was smaller, it was simpler to display it on the screen, but now that it had increased in size this became a problem.

A web form titled "Hash Family Options" with a subtitle "Select three hash functions:". It contains a list of hash functions with checkboxes: MD4, MD5, SHA1, SHA224, SHA256, SHA384, SHA512, SHA3\_256, and SHA3\_512. The checkboxes for MD5, SHA1, and SHA256 are checked and marked with green checkmarks. At the bottom of the form is a blue "UPDATE" button with a circular arrow icon to its right.

Hash Family Options	
Select three hash functions:	
<input type="checkbox"/>	MD4
<input checked="" type="checkbox"/>	MD5
<input checked="" type="checkbox"/>	SHA1
<input type="checkbox"/>	SHA224
<input checked="" type="checkbox"/>	SHA256
<input type="checkbox"/>	SHA384
<input type="checkbox"/>	SHA512
<input type="checkbox"/>	SHA3_256
<input type="checkbox"/>	SHA3_512
<div>UPDATE ↻</div>	

**Figure 5.8:** List of hash function options.

One idea was to display the entire filter at the bottom of the screen and extract a row from it and display underneath the statistics but above the check search. The extracted row needed to contain all the values; 0, 1, 2, 3 and 4. At a glance, finding such a row was not the issue. The issue was that the hash functions used to insert the Cork dataset into the Spatial Bloom Filter could be altered, figure 5.8. By using different combinations of hash functions, extracting a row from the filter with the same meaningful data became too tedious to implement.

The solution was as described above in **5.1.4.4 Check**, the entire filter was displayed, and the check results would link the appropriate cell index and be

highlighted in red. By hovering over any of the red cells, a tool-tip would display stating which of the hash function returned that Area of Interest.

## 5.3 SERVER

The web application was deployed onto Amazon Web Service. To start the demonstration on a local machine was entirely different to having it run full time on a server and not as straightforward.

```
# To run the demonstration on a local machine
> Python3 demo.py
```

**Listing 16:** Start flask application on local.

### 5.3.1 AWS Setup

The first step to creating the server was to start up a new EC2 instance on AWS. The instance needs LAMPy (Linux, Apache, MySQL, Python) stack installed to run as a web server. Since the EC2 instance is already running Ubuntu, the Linux part is already complete. Linux also has Python installed which leaves apache as the next software to be installed. MySQL is not required for this web demonstration. To complete the LAMPy stack, these are some necessary packages to install on the server included:

- apache2
- libapache-mod-wsgi-py3
- Python3-pip

For this Python web application to run, these Python packages must be installed:

- flask
- numpy
- Python3-pip

### 5.3.2 Apache

To enable Apache to serve Flask web applications, `mod_wsgi` must be enabled. `mod_wsgi` is an Apache HTTP server mod that acts as an interface between the web server and the web application. The code that `mod_wsgi` executes is stored in a `wsgi` (Web Server Gateway Interface) file that contains the code to start the web application. Python packages that had to be installed through pip also need to be imported into this file. See `demo.wsgi` code; listing 17.

```
#!/usr/bin/Python3
import sys
import flask
import numpy
sys.path.insert(0, '/var/www/demo')
from demo import app as application
```

**Listing 17:** `demo.wsgi` file contents

As the web demonstration code is in GitHub repository, it is cloned onto the EC2 instance, and updates pulled when they are available. The repo directory must be located within the `/var/www` directory for apache to serve the Flask application with the virtual host file updated to reflect this change. See the virtual host file code snippet; listing 18.

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/demo

    WSGIDaemonProcess demo threads=5
    WSGIScriptAlias / /var/www/demo/demo.wsgi

    <Directory /var/www/demo>
        WSGIProcessGroup demo
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

**Listing 18:** Virtual Host File Contents

### 5.3.3 Difference Between AWS and Local

The version of Python installed on the EC2 instance was Python 3.5 whereas on local, was Python 3.6. The only issue this created was that Python 3.5 had less hash function available in hashlib. None of the sha3 hash functions was available in the Python 3.5 hashlib, but they were in Python 3.6.

However, Python 3.5 was able to import them using the package sha3. By checking the version of Python available determined whether or not this package was imported. The sha3 hash functions were now able to use again. See listing 19.

```
import sys
if sys.version_info < (3, 6):
    import sha3
...
m = hashlib.sha3_256()
```

**Listing 19:** detect python version



## **6 Evaluation**

### **6.1 TESTING**

Testing the code is an essential part of any project. It plays a critical role in determining whether the project was the success or failure of the project. It can highlight functions incorrectly implemented as they do not return the expected result which can save time as it catches the problem early.

#### **6.1.1 Unit Testing**

Unit testing was the primary method of testing code. For every new class or every new method added to a class, these new additions had to be adequately tested.

PyCharm can create a test for each class inside the project. It can open the test file or create one if there is none detected. When creating a test file, it allows the option of creating test methods for each method within the original class.

Each unit test or the entire test class can be ran with coverage. Coverage is a highlighting tool which showcases what code was covered by the test and what code was not. This gives good feedback as to what needs to be tested next.

#### **6.1.2 Exploratory Testing**

The front-end testing was done manually. As component sizes were tweaked, the layout changed and new features added; these had to be tested from the point of view of a user. There is no set procedure to follow, and it is a small scale web application, exploratory testing is a suitable approach.

Testing involves exploring every feature of the web application to ensure that they work. For a manual test to be a success, the tester must be highly knowledgeable of the application. If the tester is not familiar with the application, the testing is incomplete as something might be missed.

### 6.1.3 Known Errors

The only known error was an issue with clearing the test data for coordinates that returned incorrect Areas of Interest. When importing the Cork dataset, the test data for incorrect values are also calculated. The test data coordinates are to demonstrate how the filter operates when it produces inaccurate results.

As the hash functions can be modified in the edit details page, if they are updated, then the Spatial Bloom Filter and all the statistics are cleared. The Cork dataset will need to be imported again using the newly selected hash functions. The test data should also be updated; however, occasionally, the previous test data is kept. This error could be rectified by clearing the Spatial Bloom Filter and importing the Cork dataset again.

As this error only occurred twice, it was difficult to determine the cause. As this information was stored in Flask's session store, the problem might be related to a caching error with the cookie. However, this has not been proven.

## 6.2 USER FRIENDLY

The goal of web demonstration was to explain to non-experts how the Spatial Bloom Filter worked. To evaluate if the demonstration was fit for purpose, non-software developers tested the web application.

User feedback on the demonstration included:

- *"clear and easy to navigate"*

- *"a clear and neat layout"*
- *"the visual aid of the map helped to understand what was being tested"*
- *"page wasn't cluttered with text and the about part had a clear layout"*

As the map of Cork aided in the understanding of where the data was coming from, a snippet of the map depicting an Area of Interest was included on the About page. It was accompanied by an explanation of where the Cork dataset came from and what the check results indicated. Keeping this information close to one another, clear and concise, increased user understanding.

## 6.3 GRAPHS

To make an informed decision on what size filter to use for the web demonstration, visual representation of selected filter statistics were created.

### 6.3.1 False Positive Probability

For the purpose of the demonstration, false positive values are wanted to produce situations where the Spatial Bloom Filter returns an incorrect result, while keeping the size of the filter at a manageable size.

By examining the graph, see appendix A.1, a filter that has 1024,  $2^{10}$ , cells is ideal with a false positive probability of 0.3898. A filter with 512 cells has false positive probability that is too high at 0.7785. A filter with 2048 cells has a false positive probability of 0.1164 but it's size is too large for demonstration purposes.

### 6.3.2 Sparsity and Hash Collisions

The ideal Spatial Bloom Filter would have a high sparsity rate to ensure minimum collisions. For the purpose of the demonstration, the sparsity was kept low to produce more hash collisions so values could be overwritten and output the incorrect Area of Interest that it belonged to.

By examining the graphs, see appendix A.2 and appendix A.3, as the sparsity increases the number of hash collisions decreases. A filter with 1024 cells was the preferred option for its false positive probability, the sparsity for a filter that size was 0.2695 and the number of hash collisions totalled 623. These values confirmed that a Spatial Bloom Filter with 1024 cells was the best choice for the demonstration.

## 6.4 TABLES

### 6.4.1 The Spatial Bloom Filter

The statistics of the Spatial Bloom filter used for the demonstration can be seen in table 6.1.

**Table 6.1:** Demonstration filter statistics.

<b>Number of cells</b>	1024
<b>False Positive Probability</b>	0.3898
<b>Sparsity</b>	0.2695
<b>Hash Collisions</b>	623
<b>Hash Family</b>	MD5, SHA1, SHA256

#### 6.4.1.1 False Positive Probability

The probability that a coordinate, when queried, will return the wrong result.

#### 6.4.1.2 Sparsity

The ratio of non zero cells in the filter to the total number of cells of the Spatial Bloom Filter.

#### 6.4.1.3 Hash Collisions

The number of collisions that occurred.

#### 6.4.1.4 Hash Family

The hash functions used in the Spatial Bloom Filter.

## 6.4.2 Areas of Interest

The properties of each Area of Interest used in the demonstration can be seen in table 6.2 while their statistics can be seen in table 6.3

**Table 6.2:** Area of Interest Properties.

AoI	Members	Cells Used	Potential Cells	Self Collision
1	217	232	651	182
2	106	177	318	50
3	21	43	63	3
4	113	296	339	43

### 6.4.2.1 Members

The number of coordinates belonging to the Area of Interest.

### 6.4.2.2 Cells Used

The number of cells used by the Area of Interest.

### 6.4.2.3 Potential Cells

The number of cells it could have used which is the member multiplied by the number of hash function, which in this case is 3.

### 6.4.2.4 Self Collision

The number of times it collided with values from the same Area of Interest and overwrote them.

**Table 6.3:** Area of Interest Statistics

<b>AoI</b>	<b>Emersion</b>	<b>False Positive Probability</b>	<b>Inter Set Error Probability</b>
1	0.4947	0.2618	0.1290
2	0.6604	0.0917	0.0391
3	0.7167	0.0121	0.0227
4	1.0000	0.0242	0.0000

#### **6.4.2.5 Emersion**

The ratio of cells set to the Area of Interest and the cells that would have been set to the Area of Interest with no collisions from higher Areas of Interest.

#### **6.4.2.6 False Positive Probability**

The false positive probability of the area.

#### **6.4.2.7 Inter Set Error Probability**

The probability of coordinates from an Area of Interest, when queried, will return a different Area of Interest.

## 7 Conclusions

### 7.1 SUMMARY AND ACCOMPLISHMENTS

The goal of this project was to explain to non-experts how a Spatial Bloom Filter operates through the use of a simple graphical web demonstration. Using test data based on the city of Cork, four Areas of Interest were marked and their set of coordinates extracted to then be inserted into the filter.

The hash family used by the Spatial Bloom filter can be edited to allow for different combinations of hash families. The results of those combinations can be observed in the filter after importing the Cork dataset as the indexes of each coordinate will have changed.

After importing the Cork dataset, display the populated Spatial Bloom Filter and the related statistics. Highlight in red, the corresponding values returned in the results when the filter has been queried. A meaningful description must accompany the result to explain how the answer was reached.

For the purpose of the demonstration, the Spatial Bloom Filter was designed to have a higher false positive probability than would be expected if used in a real scenario. This was to demonstrate how the Spatial Bloom Filter could be wrong about where the coordinate belonged; whether it be that it belonged to a different Area of Interest than it was stating or that it did not belong to any Area of Interest, but the filter claims that it does belong to one.



## **7.2 FUTURE DEVELOPMENTS**

Ideas for future plans and feature to be carried out on this project.

### **7.2.1 Edit Details**

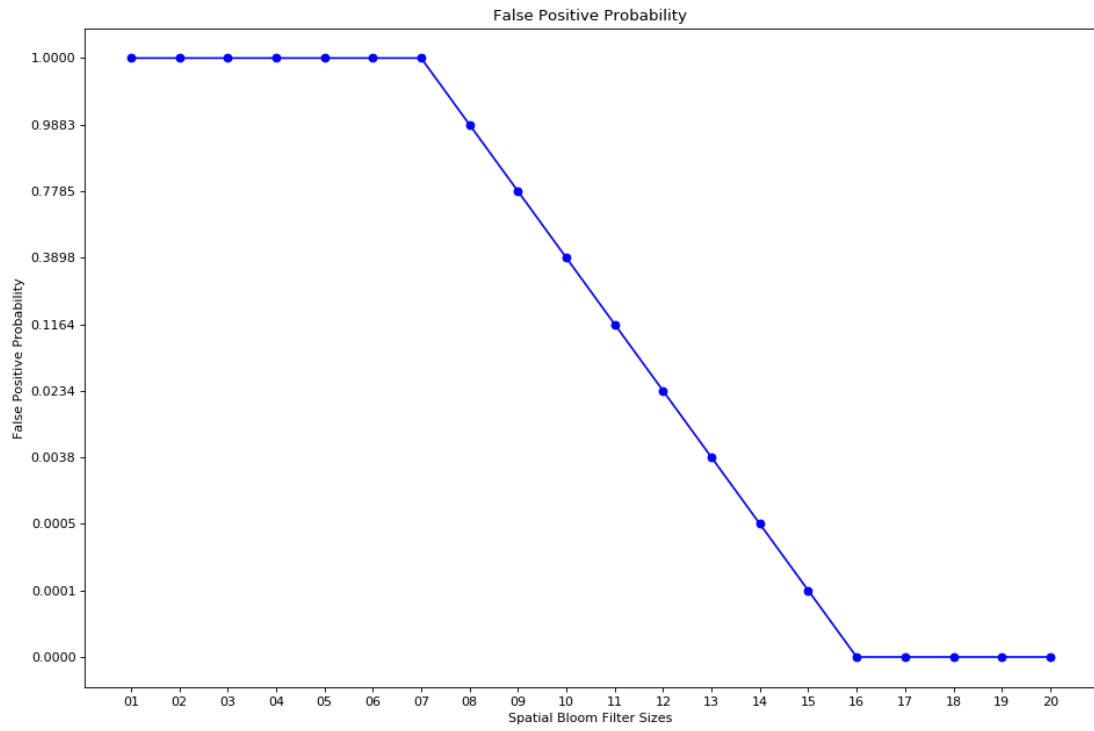
Implement a feature that would allow for the size of the Spatial Bloom Filter to be modified. By selecting various sizes for the filter, the new statistics produced would help in the explanation of what is an ideal sized Spatial Bloom Filter for the region it is used on.

### **7.2.2 Test Data**

Implement a feature that would allow the choice of the region on which the Spatial Bloom Filter is tested on. This would also allow the Areas of Interest to be chosen and the order in which they are inserted into the filter. By selecting a region that is known and familiar to the user would personalise the experience for them and could help in their comprehension of the Spatial Bloom Filter.

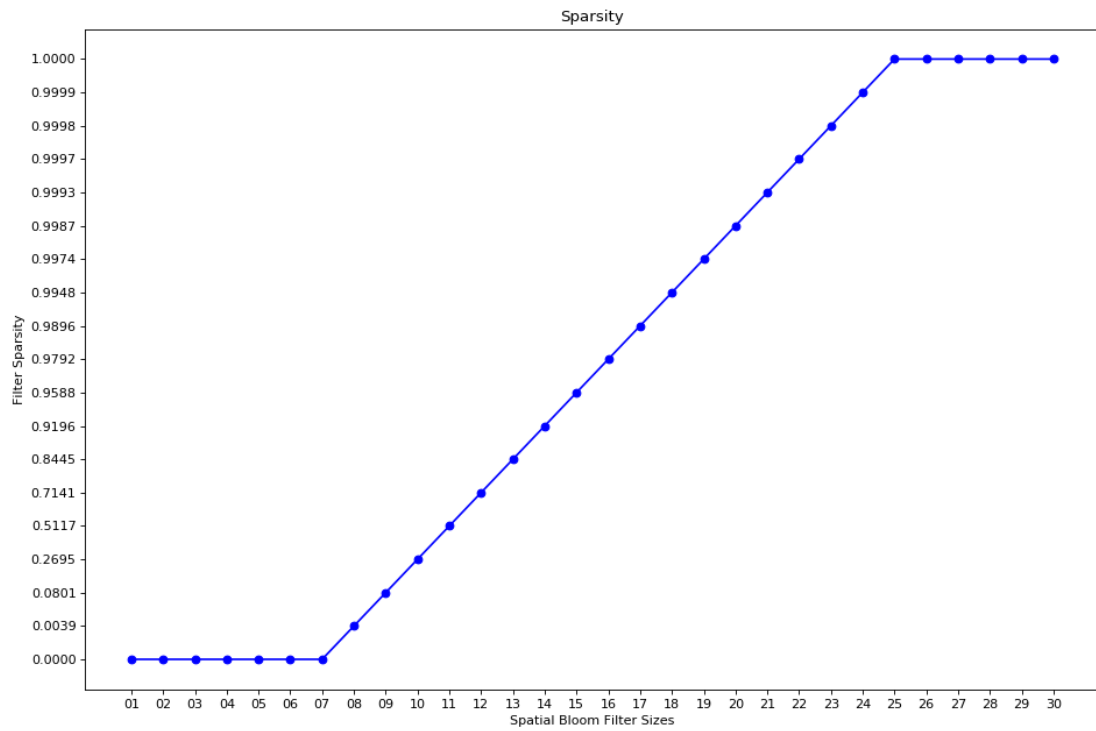
# A Appendix

## A.1 FALSE POSITIVE PROBABILITY GRAPH



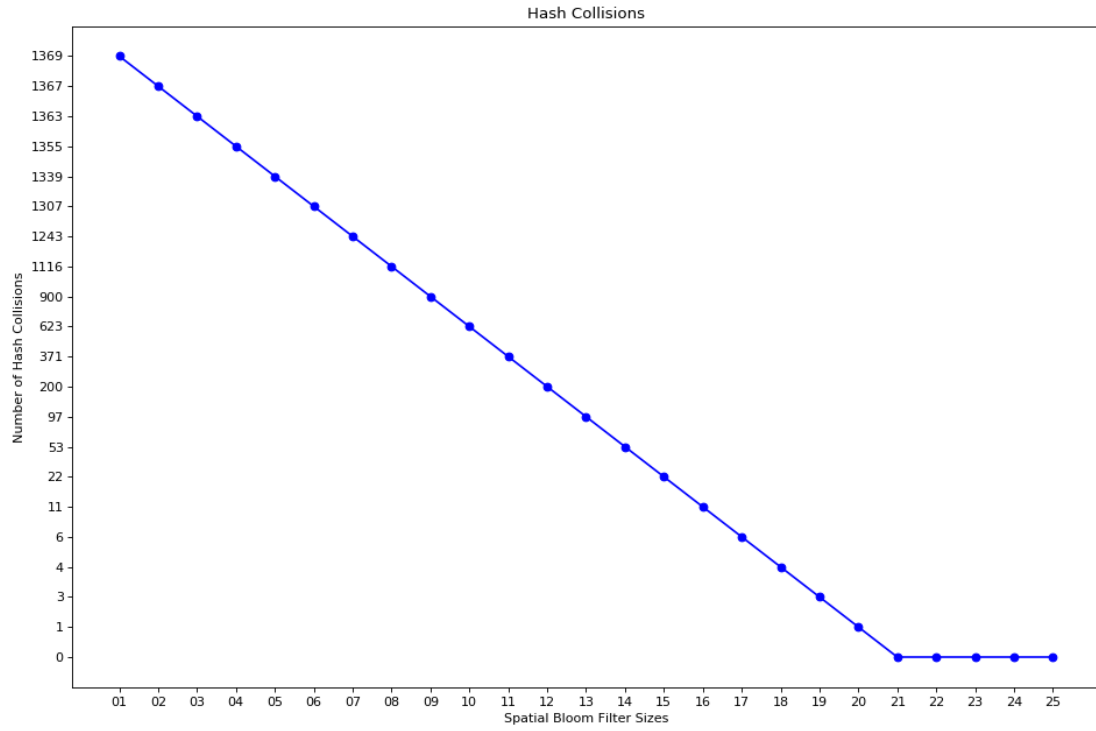
**Figure A.1:** False Positive Probability for different Spatial Bloom Filter sizes.

## A.2 SPARSITY GRAPH



**Figure A.2:** Sparsity rate for different Spatial Bloom Filter sizes.

## A.3 HASH COLLISION GRAPH



**Figure A.3:** Number of hash collisions for different Spatial Bloom Filter sizes.

## References

- [1] “Helping local businesses reach more customers.” <https://www.facebook.com/business/news/facebook-local-awareness>, Oct 2014.
- [2] D. Curran, “Are you ready? This is all the data Facebook and Google have on you.” <https://www.theguardian.com/commentisfree/2018/mar/28/all-the-data-facebook-google-has-on-you-privacy>, Mar 2018.
- [3] “Privacy Today: A Review of Current Issues.” <https://www.privacyrights.org/blog/privacy-today-review-current-issues>.
- [4] A. J. Blumberg and P. Eckersley, “On Locational Privacy, and How to Avoid Losing It Forever.” <https://www.eff.org/wp/locational-privacy>, Oct 2011.
- [5] L. Calderoni, P. Palmieri, and D. Maio, “Location privacy without mutual trust: The spatial bloom filter,” *Computer Communications*, vol. 68, 2015.
- [6] L. Calderoni, P. Palmieri, and D. Maio, “Probabilistic properties of the spatial bloom filters and their relevance to cryptographic protocols,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, p. 1710–1721, 2018.
- [7] “Bloom filters by example.” <https://l1mlib.github.io/bloomfilter-tutorial/>.
- [8] “Python 3.6.5 documentation.” <https://docs.python.org/3/>.
- [9] “Numpy.” <http://www.numpy.org/>.

- [10] “Flask.” <http://flask.pocoo.org/>.
- [11] “Materializecss.” <http://materializecss.com/>.
- [12] “Git.” <https://git-scm.com/>.
- [13] “Github | build software better, together.” <https://github.com/about>.
- [14] “Pycharm: Python ide for professional developers by jetbrains.” <https://www.jetbrains.com/pycharm/>.
- [15] “Frequently asked questions - what advantages do numpy arrays offer over python lists.” <https://www.scipy.org/scipylib/faq.html#what-advantages-do-numpy-arrays-offer-over-nested-python-lists>.