An Efficient Privacy-Preserving Satellites Collision Detection Method: Applying Private Set Intersection Using Garbled Circuit

---

A Thesis
Presented to
Established Interdisciplinary Committee for Mathematics and Computer Science
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Yuheng (Elle) Wen

May 2024

Approved for the Committee
(Mathematics and Computer Science)

Olive Franzese

Robert Chang

Acknowledgements

Thank you Olive, you are the best thesis adviser one could ever ask for, and one of the most lovely people I met at Reed. This thesis is not possible without you. Thank you Robert for the math discussions, and for showing me hard math can be done in a chilled way back in Real Analysis class.

Thank you Charlie for bringing me into computer science research. Thank you Meaw for showing me the hope of cryptography. Thank you Greg for the great classes. Thank you Alex and Adrian for your kindness as real educators.

Thanks to all my wonderful friends whom I like, and made di erent kinds of connections with during my time at Reed (in alphabetical order): Abi, Clay Steinhilber, Clay Liu, Gabe Sasu, Guangyi, Kara, Louise, Max Bennett, Sophronia, Victor, Xinlong, Xinran. My Reed is incomplete without you guys.

Thanks to my sisters back at home: Lu, Zihui, Jiangyue. Thank you for always being there for me.

Thanks to Muye, for making me a happier and stronger person.

Last but not least, mom and dad, thank you for supporting me and sending me o to see the world.

Table of Contents

Abstract

Satellite operators have high incentives to protect their satellites from collision due to the high cost of designing, building, launching and maintaining the satellites. Unfortunately, the fact that satellite owners often view the satellite trajectories as private posts a serious barrier to coordination between di erent operators for more accurate collision detection and prevention. This privacy concern becomes more apparent for satellites used for military purposes given that satellites location could reveal countries' military operations, intelligence-gathering methods, interests in specific regions of the Earth, or technology capabilities.

A 2014 report from the RAND Corporation proposed a method that enables satellite operators to calculate collision probabilities (conjunction analysis) without sharing private information about the trajectories of their satellites using cryptographic tools for the first time. Two years later, a paper optimized the implementation proposed in the RAND paper. However, even with the optimization, this method is still too slow to feasibly run it on all of the objects to detect possible collisions and thus is impractical if the operator owns multiple satellites.

In my thesis, we propose and implement a new method that is able to detect the satellites at risk of collision without revealing the location information using "fuzzy" private set intersection (PSI) and PSI with significantly reduced running time. The running time of our method is dramatically quicker and making our method much more scalable. Thus, it is able to run on every pair of first 751 satellites extracted from the most recent satellites tle data provided by Space-Track with 5.5×10^7 AND gates, where the sate of art of conjuction analysis would take 5.7×10^1 MULT gates. And each MULT gate requires di erent number of AND gates depending on the number of bits and implementations. However, the speed comes with the expense of accuracy. With di erence choice of parameters, there are di erent levels of false positives and false negatives rates. When the collision distance is 500km, there can be around 20% false negatives rates.

Chapter 1

Introduction

With the increasing number of satellites and space debris, tracking and predicting the positions of these objects has become crucial to prevent collisions. The capacity to manage and process this data, however, varies significantly among nations, with only a few powerful countries possessing advanced monitoring capabilities. These nations dominate in both the volume of satellites they manage and their control over SSA data. The current methods of collision avoidance primarily depend on central databases like the Space Surveillance Network, which pose risks of data manipulation and provide less accurate data compared to what satellite operators can track independently. This reliance on potentially compromised and less precise data underscores the need for improved systems of data sharing among satellite operators.

To address these challenges, a novel approach involving secure Multiparty Computation (MPC) was proposed by RAND Corporation in 2014. MPC is a cryptographic tool that allow parties without trust to collaboratively compute functions. This method allows satellite operators to calculate collision probabilities without having to share the private trajectory data of their satellites, ensuring privacy and reducing reliance on potentially unreliable central databases. It does this by converting the probability calculation to binary gates and applying the standard garbled circuit protocol to each gate. This approach is seen as a potential solution to the limitations of current practices by enabling more accurate and secure data sharing. Despite its potential, implementing this secure method is computationally intensive. This is due to the fact that conjunction analysis requires the computation of complex integrals, exponential functions, and matrix operations. These operations demand the translation into millions of binary gates. Therefore, this method is not practical for operators managing multiple satellites as it is infeasible to apply this method on every pair of satellites. A faster method would make satellite collision detection

more schalable, enable real-time data processing, allow satellite operators to dynamically adjust satellite paths, and thus significantly enhance the ability to manage and mitigate collision risks.

The first method proposed by this thesis employs fuzzy Private Set Intersection (PSI). Private Set Intersection is a cryptographic technique that allows multiple parties to compare their private datasets and find common elements without revealing any other information about the datasets. Fuzzy PSI extends this concept by allowing for the identification of common elements that are similar, but not exactly identical, based on certain predefined rules or similarity metrics. This modification makes it particularly useful in scenarios where exact matches are rare or impractical. In MPC, the number of AND gates is often used as a metric for computational cost as AND gate is often the bottleneck in the speed of MPC algorithms. This method uses around 7500,0000 AND gates to compute colliding satellites if two parties each owns 500 satellites. The second method using LSH and PSI uses only 2500,0000 AND gates to compute colliding satellites for the same number of satellites. The second method loses some degree of accuracy in compensation for the speed for di erent choice of the parameters. When the collision distance is 500km, there can be around 20% false negatives rates.

The primary contribution of this method is that it does not require the computation of conjunction analysis calculations, which involve multiple steps of arithmetic operations. Instead of computing the probability of collision, this method directly returns the possible satellites at risk of collision when provided with the collision distance parameter. While avoiding the conjunction analysis calculation gains us speed, it also means that we choose proximity as a less precise metric of possibility of collision instead of trajectories. Nevertheless, this thesis sheds light on a promising direction for another method to establish a cooperative space data sharing prototype.

The first section of background chapter overviews the current spatial situation, the primary method for satellite collision avoidance, and the two main motivations for using MPC in collision detection. The next section introduces the existing MPC method for detecting collision and its limitation. And the last section covers the background knowledge of the cryptographic concepts and tools that are used in the method proposed by this thesis. The methods chapter in the thesis details two approaches for satellite collision analysis. Initially, it presents a more accurate but relatively slow method. Subsequently, it introduces a faster method that trades o some accuracy for increased speed, making it more viable for real-time satellite management. The results chapter shows the running time and accuracy of the two proposed methods.

Lastly, the conclusion chapter recaps the contribution of this thesis and discusses the possible concerns in practices and the future direction of this research.

Chapter 2

Background

2.1 Satellites in the Space

The current situation of space surrounding Earth has an abundance of satellites, debris, and various spatial objects. This congestion underscores the critical need to keep track of objects in orbit to avoid collisions. This knowledge is called Space Situational Awareness (SSA). However, the capacity to gather and process data required for e ective SSA is unevenly distributed among nations. While some powerful countries have access to advanced observational equipment that allows them to have more spatial objects' location data, less resourceful nations often have fewer SSA data. In addition, the geolocational distribution of advanced observational equipment is significantly skewed towards more powerful nations, particularly those with a global military presence, such as the United States. According to the US Department of Defense, by December 2023, US has a total of 226,762 overseas active duty troops in 176 countries while there are only 237 countries in the world as listed by CIA. The US, with its extensive network of military bases around the world, is able to combine observational data from multiple perspectives. This global coverage allow it to construct an exhaustive set of observations of spatial objects. However, very few other nations have this capability.

The United States, China, United Kingdom, and Russia currently lead in space capabilities and the collection of space data. Their dominance not only reflects in the volume of satellites they operate but also in the control and dissemination of space situational data. In particular, U.S., China, and Russia lead the number of satellites in military uses. As shown in Figure 2.1 and Figure 2.2, some countries own over thousands of satellites while others own just one or two.

Presently, the primary method for collision avoidance for satellites involves reliance

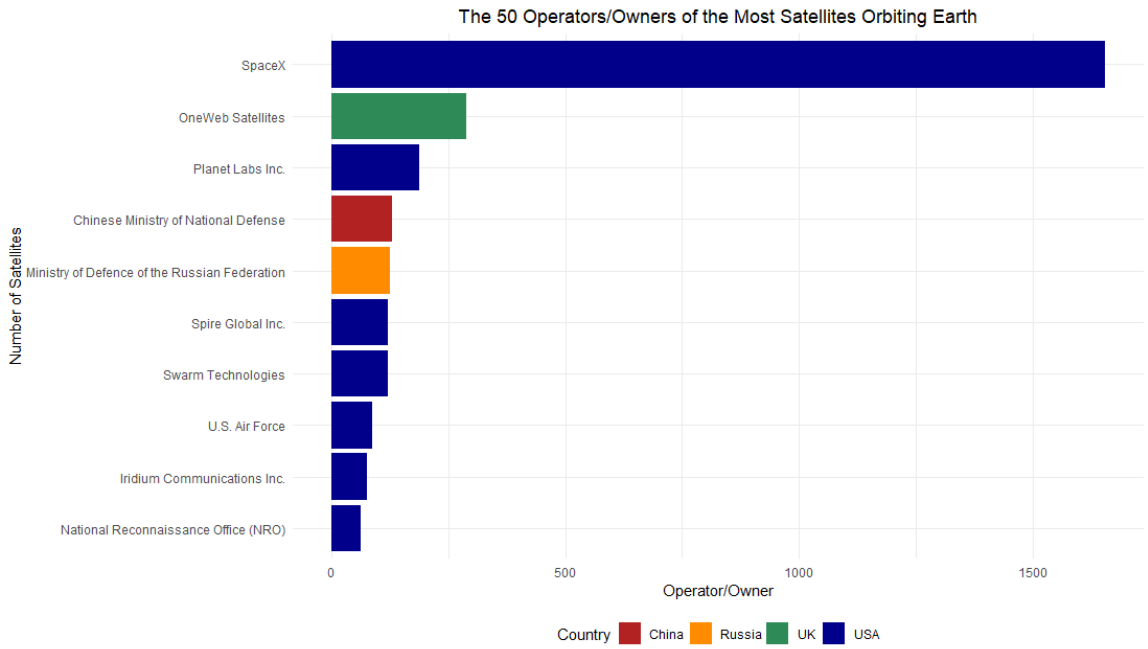

Figure 2.1: The top 10 operator owners according to DEWESoft published in February 2023. Note that China, Russia, and U.S. own the most operators in military uses.

on a handful of central database provided by a more resourceful party such as Space Surveillance Network managed by U.S. Strategic Command, Space-Track.org, or CelesTrak. Satellites filter through the database to identify objects in close proximity and then conduct pairwise comparisons and calculations to assess collision risks.

This status quo has two main drawbacks. First, it poses a security risk, as it heavily depends on data which might be manipulated or withheld by countries for strategic reasons. For instance, a country might have incentives to misrepresent information about a spy satellite to covertly approach foreign satellites or assets. Second, the tracking data obtained by the central database is acquired through observations, and thus is significantly less accurate than the active tracking information held by each satellite's operator. This is because he most accurate information comes from onboard instrumentation, but this information is available only to the satellite operator. Since satellite operators maintain accurate tracking information for only their own satellites, sharing this higher-fidelity information between satellite operators could provide significantly better tracking information than what can be obtained by a central database.

As an example, a comparison of cooperative and non-cooperative tracking data

Rank	Country	Total Number of Satellites
1	USA	2804
2	China	467
3	United Kingdom	349
4	Russia	168
5	Japan	93
6	India	61
7	Canada	57
8	Germany	47
9	Luxembourg	40
10	Argentina	34
11	France	31
12	Spain	24
13	Italy	21
14	Israel	19
15	South Korea	18
16	Brazil	16
17	Netherlands	16
18	Finland	15
19	Australia	14
20	Saudi Arabia	13
21	Taiwan	13
22	United Arab Emirates	13
23	Switzerland	13
24	Singapore	11
25	Turkey	9
26	Indonesia	8
27	Norway	8
28	Mexico	8
29	Thailand	7
30	Kazakhstan	6
31	Algeria, Belgium, Greece, Sweden	5
32	Denmark, Egypt, Malaysia, Vietnam	4
33	Czechia, Morocco, Nigeria, Pakistan, South Africa	3
34	Azerbaijan, Belarus, Ethiopia, Lithuania, Slovenia, Venezuela	2
35	Austria, Bangladesh, Bolivia, Bulgaria, Chile, Colombia, Ecuador, Estonia, Hungary, Iran, Iraq, Jordan, Kuwait, Laos, Mauritius, Monaco, Nepal, New Zealand, Paraguay, Peru, Qatar, Sri Lanka, Sudan, Tunisia, Turkmenistan, Ukraine	1

Figure 2.2: The countries with the most satellites published by DEWESoft in February 2023

for Global Positioning System satellites found that cooperative tracking data reduced mean positional error by 88 percent (1). This means the mean error for satellites tracking using non-cooperative tracking is 7.267km while the cooperative tracking is only 0.872 km. Thus, satellite operators could acquire much more accuracy regarding collision detection if they all cooperated by sharing the location information from on-board instrumentation. However, even if a operator who doesn't want to rely on the central public database wants to cooperate directly with other operators, privacy

concerns still present a serious barrier, as operators often view the trajectories of their satellites as private, and refuse to share this private information with others. Using techniques from cryptography, we can improve accuracy while keeping the data private.

2.2 Multi-Party Computation for Conjunction Analysis

One cryptographic tool that helps improve accuracy and privacy is secure Multiparty Computation (MPC). MPC is a broad area of research in cryptography which has attracted the attention of many researchers. The goal of MPC is to allow a group of distrustful parties, $P = \{P_1, P_2, \ldots, P_N\}$, to perform joint computations on private inputs while ensuring input privacy and output correctness.

Andrew Yao's early 1980s work, particularly the Garbled Circuits Protocol (2), laid the groundwork for MPC. It is not until entering 2000s' that this field transitioned from theoretical entertainment to practical application due to algorithmic and computing power improvements. The first MPC protocols proposed by Yao is a generic protocol–it can be used to compute any discrete function that can be represented as a fixed-size circuit.

In the literature of MPC, three primary adversary models are commonly discussed:

> The *Passive* or *Semi-Honest Adversary*, where corrupted parties adhere to the protocol but attempt to glean extra information from the acquired data.
>
> The *Active* or *Malicious Adversary*, where corrupted parties may stray from the protocol for their benefit.
>
> The *Mixed Adversary*, where it encompasses scenarios where some parties are corrupted passively and others actively.

This thesis considers the semi-honest adversary as a precursor to the stronger malicious model. In addition, since all parties benefit from following the protocol, it is reasonable for us to assume they don't want to stray from the protocol.

A novel approach for satellites collision detection that uses MPC has been proposed in 2014 (3). This report proposed using secure Multiparty Computation (MPC) to allow satellite operators to calculate collision probabilities (conjunction analyses) without sharing private information about the trajectories of their satellites.

For two parties A and B, each of them has a satellite and four private parameters that describe the trajectory of that satellite. Then they input the four parameters to a secure version of conjunction analysis algorithm. The algorithm outputs the probability of collision for the two satellites to each party. See Figure 2.2 for the private inputs given to the algorithm.

Position: $P_a, P_b \in \mathbb{R}^3$
Velocity: $V_a, V_b \in \mathbb{R}^3$
Error: Covariance matrices $C_a, C_b \in \mathbb{R}^{3\times3}$
Radius: $R_a, R_b \in \mathbb{R}$

Figure 2.3: Private inputs from each satellite operator.

The conjunction analysis calculation in clear is discussed in more detail in Appendix B. At a higher level, conjunction analysis is a computation of the collision probability of two satellites involving complex matrix operations, integral calculations, and exponential function calculations.

The report then proposed that to make this in-clear algorithm secure, we could either use Yao's garbled circuit or Goldreich, Micali, and Wigderson (GMW) protocol to convert the function into secure binary circuits. In other words, for each arithmetic operation, there is a corresponding binary gate that compute the results securely.

The report did not actually implement the secure conjunction analysis. They approximated its time complexity by estimating the number of binary gates used during the computation. This is because the actual running time depends on many factors including hardware system and the desired numerical precision of the calculation. In digital computing, each function can be expressed as a binary circuit, meaning that functions are decomposed into sequences of AND and OR gates for processing by the central processing unit. Likewise, functions can be reformulated using ADD and MULT gates, forming what is known as an arithmetic circuit. The duration of a single gate operation, when multiplied by the total number of gates required to compute the function, o ers a metric for estimating the computation time for any function. For one pair of satellites, it takes 100 million binary gates to calculate its collision probability securely.

In a later work published in 2016, they optimized the circuit and were able to use 101574 MULT gates and 267002 total gates to calculate the collision probability securely for one pair of satellites (4).

However, because of the fact that the doing secure conjunction analysis requires many precise computations of complex arithmetic operations, this proposed method

is very slow. And thus in the more majority cases where one party owns multiple satellites, using this method to calculate the collision probability for every pair of satellites becomes extremely inefficient as the slowness of secure conjunction analysis makes it infeasible to perform this method on all possible pairs of satellites. Therefore, this method is impractical in most of the cases due to its running time limitation. A faster approach needs to be proposed.

2.3 Private Set Intersection

In this thesis, we use Private Set Intersection (PSI) to make a method for satellites collision detection with improved efficiency while guaranteeing the privacy of the satellites location information.

Private Set Intersection (PSI) is a specific problem within MPC that allows two or more parties to introduce their private sets as inputs and compute the intersection while nothing else can be inferred.

Based on the size of the sets, PSI can be classified into two categories: balanced set size and unbalanced set size. This is because di erent factors needs to be considered when designing and evaluating PSI protocols given the two scenarios. Balanced set size is the scenarios when all parties have relatively same number of elements in their private set. And unbalanced set size is the scenarios when some of the parties have larger set than the others. In this thesis, we consider balanced set. This is because the motivation of this work is to allow less resourceful countries to cooperate together, and they have similar number of satellites as shown in Figure 2.2.

Fuzzy PSI is a variation of PSI in the way that it allows for the identification of common elements that are similar, but not exactly identical, based on certain predefined rules or similarity metrics. This concept was firstly introduced by Freedman et al, who gives a protocol for Hamming-distance (over tuples of strings) (5). It allows parties to learn which of their points are within distance ϵ. In the satellites collision detection case, fuzziness refers to individual coordinates of the parties that are close but not necessarily equal. This modification makes it particularly useful in the satellite collision detection scenario as most of the time, satellites are at risk of colliding when their coordinates are close and not necessarily when they exactly match.

There are di erent ways to solve the PSI problem (6). Over the past years, intensive research have been done designing custom protocols for PSI based on homomorphic encryption such as the works of (7; 8) and other public-key techniques (9; 10). However, Garbled circuits is still often chosen to implement PSI algorithm

due to its flexibility and speed (11). Specifically, it is shown that a careful application of garbled circuits leads to solutions that can be competitive with the fastest custom protocols (11). And because it is generic, one can simply create a circuit for the desired function by utilizing available software packages (12; 13; 14) for building garbled-circuit protocols instead of developing and implementing an entirely new protocol.

In this thesis, we use a garbled circuit implementation of fuzzy PSI and PSI to allow operators to detect satellites collision collaboratively with faster speed.

Chapter 3

Method

3.1 Overview

In this study[1], we explore a scenario involving multiple parties, each denoted as $B_1, B_2, ..., B_i$, where i represents the total number of parties involved. Each party possesses a private dataset comprising tuples in the form of (coordinate, object), with the objective of generating a new dataset. This new dataset should include data points where the coordinates are within an ϵ distance from the private sets of other parties, meaning those points are at risk of collision. This way, parties are able to detect satellites collision.

Specifically, in the context of space situational awareness, B_i symbolizes countries equipped with limited observatory capabilities. The private dataset of B_i encompasses information about space objects that are known to B_i but may not be known to other parties. To simplify our experimental setup, we focus on a scenario with only two parties ($i = 2$), although the methodology is scalable to a multi-party setting.

The satellite location data were partitioned into two groups, each representing the private dataset of one party. To identify the intersections of these datasets, which indicate potential collision courses between space objects, we employed the Private Set Intersection (PSI) algorithm. PSI was implemented using emp toolkit. Emp is an efficient multi party computation toolkit written in C++ based on garbled circuit developed by Wang Xiao in Northwestern University. Given the nature of space object collisions, where the satellites that are at risk of collision mostly don't have exact coordinate match, we initially applied a secure fuzzy PSI computation using garbled circuits.

---

github repository link for the codes

However, the practical running time associated with direct fuzzy PSI computations posed significant challenges. To address this, we adopted a dimensionality reduction technique known as locality-sensitive hashing (LSH), e ectively reducing the running time.

3.2 Data acquisition

The satellite coordinates for this study were acquired from Space-Track.org, a database managed by the Science Applications International Corporation (SAIC) in Virginia, which specializes in government and information technology services. Space-Track.org o ers complimentary access to Space Situational Awareness Data to the international space community. The dataset includes the most recent Two-Line Element (TLE) sets for all tracked objects updated within the preceding 30 days, available in the "full catalog" section of the Space-Track.org website. TLE is a standard data format used to describe the location of Earth-orbiting object using a list of orbital elements for a given point in time (epoch). Using a suitable prediction formula, the State (position and velocity) at any point in the past or future can be estimated to some accuracy. An example of TLE data for an object is showed below in Figure 3.1 and its explanation is showed below in Figure 3.2. To transform the TLE data into Cartesian

```
ISS (ZARYA)
1 25544U 98067A   04236.56031392  .00020137  00000-0  16538-3 0  9993
2 25544  51.6335 344.7760 0007976 126.2523 325.9359 15.70406856328906
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890| reference|number line|
         1         2         3         4         5         6         7|
```

Figure 3.1: Example TLE data

coordinates (x, y, z), I utilized the Python package Skyfield. This package processes TLE files by employing the SGP4 satellite propagation model (15). I partitioned the coordinates into two files. I use the first half of the data to simulate objects owned by party A and the second half owned by party B.

3.3 Fuzzy PSI

We first implement a fuzzy Private Set Intersection (PSI) methodology employing garbled circuits. We try this method first because the definition of fuzzy PSI aligns

LINE 0		
Columns	*Example*	*Description*
1-24	ISS (ZARYA)	The common name for the object based on information from the Satellite Catalog
LINE 1		
Columns	*Example*	*Description*
1	1	Line Number
3-7	25544	Satellite Catalog Number
8	U	Elset Classification
10-17	98067A	International Designator
19-32	04236.56031392	Element Set Epoch (UTC) *Note: spaces are acceptable in columns 21 & 22
34-43	.00020137	1st Derivative of the Mean Motion with respect to Time
45-52	00000-0	2nd Derivative of the Mean Motion with respect to Time (decimal point assumed)
54-61	16538-3	B* Drag Term
63	0	Element Set Type
65-68	999	Element Number
69	3	Checksum
LINE 2		
Columns	*Example*	*Description*
1	2	Line Number
3-7	25544	Satellite Catalog Number
9-16	51.6335	Orbit Inclination (degrees)
18-25	344.7760	Right Ascension of Ascending Node (degrees)
27-33	0007976	Eccentricity (decimal point assumed)
35-42	126.2523	Argument of Perigee (degrees)
44-51	325.9359	Mean Anomaly (degrees)
53-63	15.70406856	Mean Motion (revolutions/day)
64-68	32890	Revolution Number at Epoch
69	6	Checksum

Figure 3.2: TLE format explanation

with our goal of detecting satellites collision naturally. To recap, fuzzy PSI allows the parties to learn which of their points are within distance ϵ of the private points owned by another party. In our case, the output of the fuzzy PSI is exactly the set of satellites that are at risk of collision. This approach aims to identify pairs of data points from two distinct datasets, where the cumulative distance across all dimensions (Manhattan distance) between each pair is within a predefined threshold, ϵ.

Upon initiating the comparison, the algorithm evaluates each pair of data points, one from each party's dataset. It calculates the distance for each corresponding dimension and sums these values. A data point pair is included in the output set if and only if the total summed distance being less than or equal to ϵ. Formally, for two points (x_1, y_1, z_1) and (x_2, y_2, z_2), inclusion criteria is defined as:

$$|x_1 \quad x_2| + |y_1 \quad y_2| + |z_1 \quad z_2| \leq \epsilon.$$

The algorithm is described in 1. We run this algorithm on a benchmark dataset

Algorithm 1 Fuzzy 3D PSI

Input: elements, distance, party
Output: intersection

```
1: procedure FuzzyPSI(elements, distance, party)
2:     intersection ← an empty list of type Bit
3:     for i ← 0 to length(elements)  1 do
4:         for j ← 0 to length(elements)  1 do
5:             a ← Integer(elements[i][0], ALICE) ▷ Compute from first party's data
6:             b ← Integer(elements[j][0], BOB) ▷ Compute from second party's data
7:             di _x ← a  b
8:             abs_di _x ←  BS(di _x)
9:             c ← Integer(elements[i][1], ALICE) ▷ Compute from first party's data
1 :            d ← Integer(elements[j][1], BOB) ▷ Compute from second party's data
11:            di _y ← c  d
12:            abs_di _y ←  BS(di _y)
13:            e ← Integer(elements[i][2], ALICE) ▷ Compute from first party's data
14:            f ← Integer(elements[j][2], BOB) ▷ Compute from second party's data
15:            di _z ← e  f
16:            abs_di _z ←  BS(di _z)
17:            sum ← abs_di _x + abs_di _y + abs_di _z
18:            inside ← distance ≥ sum
19:            intersection.append(inside)
2 :    return intersection
```

but it turns out that the running time is still not ideal for the purpose of quick collision detection. The detail of the running time is shown in the result section. Then, in order to optimize the running time, we try another method.

3.4 LSH

Locality-Sensitive Hashing (LSH) is a class of functions used to reduce dimensionality while preserving the relative distance between points.

De nition 3.4.1 (LSH Family)**.** Let $\mathcal{M} = (M, d)$ be a metric space. Given a threshold $r > 0$, an approximation factor $c > 1$, and probabilities $p_1 > p_2$, a family $\mathcal{F}$ of hash functions $h\colon \mathcal{M} \to S$ is said to be an **LSH family** if it satisfies the following conditions. For any two points $a, b \in \mathcal{M}$ and any h chosen randomly from $\mathcal{F}$,

if $d(a, b) \leq r$, then $h(a) = h(b)$ with probability at least p_1; and

if $d(a, b) > cr$, then $h(a) = h(b)$ with probability at most p_2.

In this thesis, we work with a LSH family on $\mathcal{M} = (\mathbb{R}^3, \|\cdot\|)$ consisting of random projections. More specifically, for a given input $\vec{v} \in \mathbb{R}^3$, we compute its dot product with a Gaussian random vector $\vec{x} \in \mathbb{R}^3$, and further quantize the result into a set of hash bins. The intention is that nearby points in the original space $\mathcal{M}$ fall into the same bin, while faraway points fall into di erence bins.

To describe the hash functions mathematically, denote by $\lfloor \cdot \rfloor$ the floor operation, w the width of each quantization bin, and b a random scalar uniformly chosen from between $[0, w]$ that is kept the same for all points within single round of hashing. We define

$$h(\vec{v}) = h^{\vec{x},b}(\vec{v}) = \left\lfloor \frac{\vec{x} \cdot \vec{v} + b}{w} \right\rfloor. \tag{3.1}$$

We perform 3 times of hashing for one coordinate $\vec{v}$ and get three hash values for this coordinate $\vec{v}$. Next, we apply standard PSI computation based on garble circuit on the hashed values. The PSI algorithm simply compare each pair and output the ones that are the same.

Algorithm 2 LSH + PSI

Input: coordinates, distance, party
Output: intersection

```
1: procedure LSH_PSI(coordinates, distance, party)
2:     elements ← LSH(coordinates)
3:     intersection ← an empty list of type Bit
4:     for i ← 0 to length(elements)   1 do
5:         for j ← 0 to length(elements)   1 do
6:             a ← Integer(elements[i], ALICE)     ▷ Compute from first party's data
7:             b ← Integer(elements[j], BOB)     ▷ Compute from second party's data
8:             equal ← a == b
9:             intersection.append(equal)
1 :    return intersection
```

3.5 LSH analysis

We use LSH to project the 3D coordinates of the satellites onto 1D hashes, preserving the relative distance between the satellites with high probability in the hope of decreasing the computing time. We demand the following.

(1) For any pair of nearby points $\vec{v}, \vec{w} \in \mathbb{R}^d$, there is a high probability P_1 that they

fall into the same bucket:

$$\Pr(h(\vec{v}) = h(\vec{w})) \geq P_1 \quad \text{whenever} \quad \|\vec{v} \quad \vec{w}\| \leq R_1. \tag{3.2}$$

(2) For any pair of nearby points $\vec{v}, \vec{w} \in \mathbb{R}^d$, there is a low probability $P_2 < P_1$ that they fall into the same bucket

$$\Pr(h(\vec{v}) = h(\vec{w})) \leq P_2 \quad \text{whenever} \quad \|\vec{v} \quad \vec{w}\| \geq cR_1 = R_2. \tag{3.3}$$

In the above, $\|\cdot\|$ is the ℓ_2 vector norm (i.e., the Euclidean distance) and $R_2 > R_1$.

Now we want to calculate the probability of two points $\vec{v}$ and $\vec{w}$ get hashed into the same value, i.e., $\Pr(h(\vec{v}) = h(\vec{w}))$. (Recall (3.1) for the formula of the hash function.) The dot product takes the vector v from higher dimension to a real line. Division of w and the floor function does the quantization part. For the hashes of $\vec{v}$ and $\vec{w}$ to collide, two conditions must be satisfied:

$|\vec{x} \cdot \vec{v} \quad \vec{x} \cdot \vec{w}| < w$; and

The end points of bins do not fall between $h(\vec{v})$ and $h(\vec{w})$.

So we can transfer calculating the probability of the hashes of $\vec{v}$ and $\vec{w}$ colliding to calculating the probability of the two conditions are both satisfied. Since the two events are independent, the probability of the two events are both satisfied is just the multiplication of the probability of the two events happen individually.

For the first condition, since $\vec{x}$ is drawn from the standard Gaussian distribution, from direct computation we can get $\vec{x} \cdot \vec{v} \quad \vec{x} \cdot \vec{w}$ has the same distribution as $\|\vec{v}$ $\vec{w}\|\vec{z}$ where $\vec{z}$ is drawn from the same distribution. More generally, this fact is true for all random variables that are p-stable and thus this method also works if we sample $\vec{x}$ from other p stable distributions. See appendix A for examples and further discussion. Thus, the probability of the first condition is the same as the probability of $\|\vec{v} \quad \vec{w}\|\vec{z} < w$, and further equals the probability of $\vec{z} < \frac{\|\vec{v} \quad \vec{w}\|}{w}$. Recall that $\vec{z}$ is drawn from normal distribution, so the pdf of the first condition is the same as the pdf of a normal distribution. And the probability of the first condition is the pdf of normal distribution integrating from 0 to $\frac{\|\vec{v} \quad \vec{w}\|}{w}$.

For the second condition, we first calculate the probability of the the dividers of bins fall between $\vec{v}$ and $\vec{w}$, and it is just the distance of the points divided by the bin width and integrates from 0 to w. This is because the o set b for each point is

sampled in uniform from 0 to w, and thus the probability of the bin falls between the two points is uniform. Putting the two conditions together, and let $u = \|\vec{v} \quad \vec{w}\|$, we get the probability of $\vec{v}$ and $\vec{w}$ hash to the same value be,

$$\int_{x=}^{w/u} f_2(x) \left(1 \quad \frac{xu}{w}\right) dx. \tag{3.4}$$

where f_2 is the pdf of normal distribution. Note that we did a change of variable trick to combine the second condition and the first condition to be inside a single integral. Let us change the variable of integration to $t = xu$, yielding

$$\int^{w} \frac{1}{u} f_2 \left(\frac{t}{u}\right) \left(1 \quad \frac{t}{w}\right) dt \tag{3.5}$$

To make the probability close to 1 when v_1 and v_2 are very likely to be hashed into the same value, we multiply by 2. So finally we get,

$$\Pr(h(\vec{v}) = h(\vec{w})) = 2 \int^{w} \frac{1}{u} f_2 \left(\frac{t}{u}\right) \left(1 \quad \frac{t}{w}\right) dt \tag{3.6}$$

For k rounds of hashing, collision is defined by hash values being the same for all k, so the resulting probability is

$$[\Pr(h(\vec{v}) = h(\vec{w}))]^k.$$

Chapter 4

Results

As described in the method section, the satellites location data is obtained from Space-Track.org and converted to coordinates format using Skyfield. We use the first half of the data to simulate private satellites locations of party A, and the second half of the data to simulate private satellites locations of party B. We utilize actual satellite location data to enhance the testing of our method, thereby achieving more realistic and accurate results. To give a better sense of how these satellites distributes in the space, I use simple randomly sampling without replacement in uniform distribution to select points from party A and party B shown in Figure 4.1 below.

We aim to evaluate the running times and performance of the two methods outlined in the Methods section. In our secure computation approach, which is based on garbled circuits, the running time is predominantly constrained by the AND gates; thus, we use them as a proxy for assessing running time. To estimate this, we calculate the number of AND gates utilized during the secure computation benchmarks of Fuzzy PSI and simple PSI.

For the Fuzzy PSI benchmark, we generate it by randomly creating 3D points uniformly distributed between 1 and 100 for a given set size. We are able to use random points instead of real data because the runtime is independent of the content of the input and depends only on the size of the input. The benchmarks for LSH and simple PSI are generated in a similar manner. It's important to note that LSH merely hashes each point three times, making its running time linear and negligible for large set sizes compared to the second computation phase (PSI). Therefore, we represent the time complexity of the entire method—first applying LSH and then running 3D PSI—using the time complexity of 3D PSI alone. Figure 4.2 shows the running time comparison between the two methods. The graph shows that the method using LSH and simple 3D PSI has running time much smaller than Fuzzy PSI. It is noteworthy

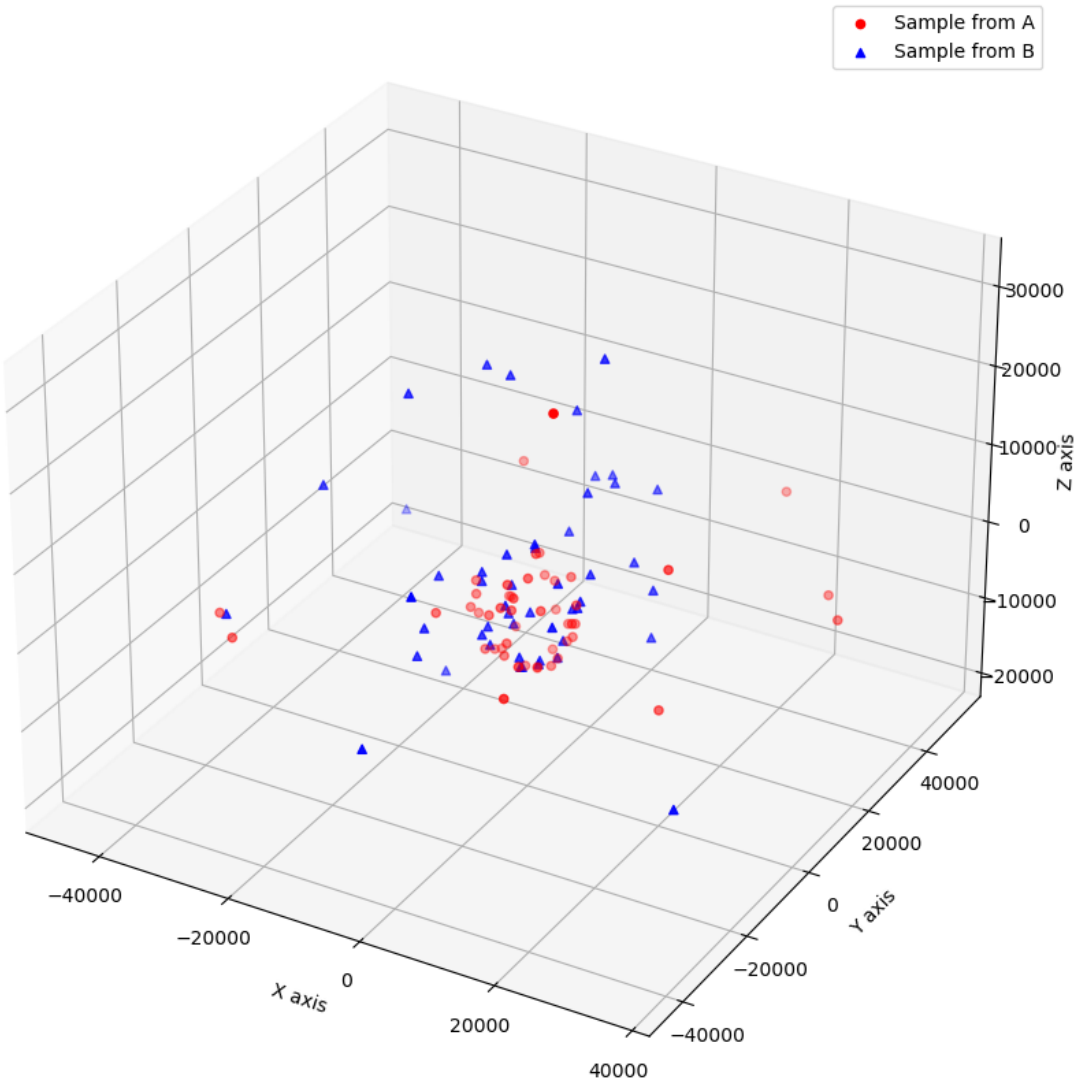

Figure 4.1: The red circles are sampled from satellites of party A and the blue triangles are sampled from satellites of party B. The scale is km.

that using 3D PSI means using three rounds of hashing in the LSH phrase. It is also possible to use just one round of hashing, and thus resulting a 1D PSI. In that case, the running time of LSH + PSI method is much more faster.

Next, we want to compare the performance of the two methods. We use the number of false positive and false negative as metrics for accuracy. A pair of points is a false positive if it is reported as a collision pair but their distance is further than the specified collision distance. A pair of points is a false negative if it is within the collision distance but it is not reported as a collision pair. For Fuzzy PSI, the number

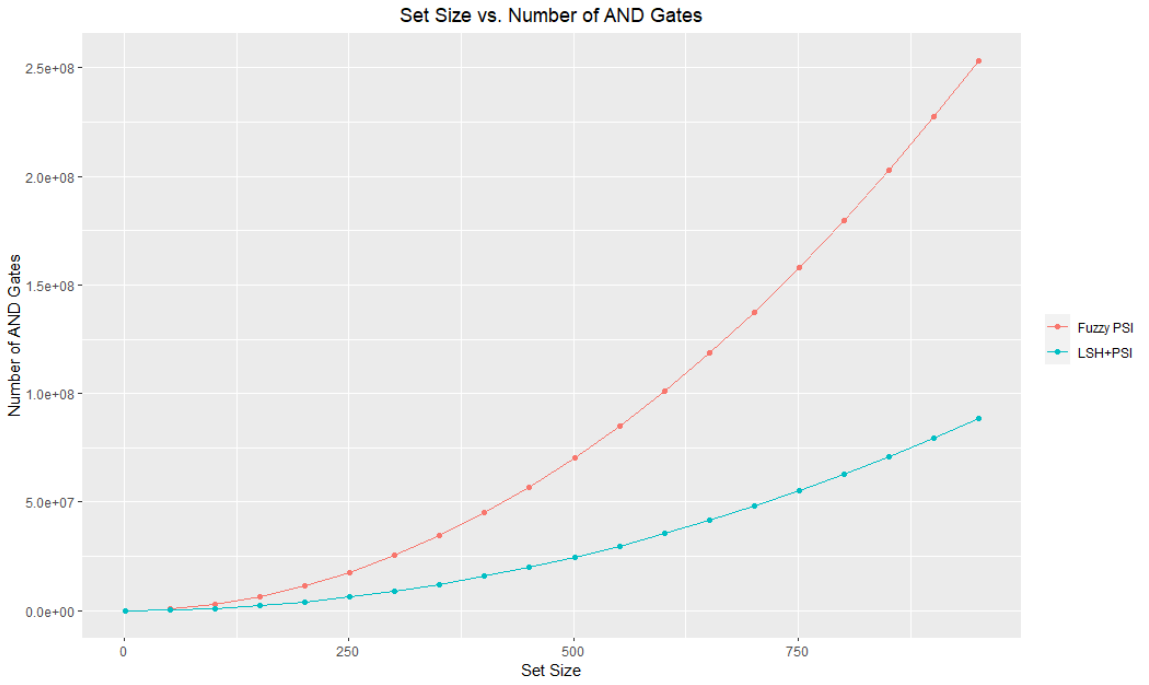

Figure 4.2: The time complexity of Fuzzy PSI and PSI benchmarks estimated using the number of AND gates.

of false positive and false negative are both 0 because our algorithm evaluates all pairs of points and returns the ones that have distance between the specified ϵ and exclude the points that are not.

For LSH and simple PSI method, since the PSI we used visit each pair of hash values, there is no room for inaccuracy in this part, and the number of false positives and negatives are purely results from LSH. As shown in the method section, the probability of collision depends on the distance between two points, the bin width, and the number of k-products. This means that the accuracy of this method depends on what values of collision distance, k, and bin width we choose.

We choose the collision distance to be 500km because results show that a satellite with a detection range of 500km would be able in more than 80% of the cases to observe a high-risk object twice and for at least 10s before the potential collision (16). Therefore, we want the collision probability for points that have distance within 500km to be large and for distance greater than 500km to be small.

To choose the parameters, we pick a distance and variate the bin width. For each bin width, we calculate the collision probability using the closed formula described in the method section to get a general sense, and then go through each pair of hashed

points to count the total number of false positive and false negative pairs in our simulated dataset. The collision probabilities for $k = 1$ and $k = 3$ are shown in Figure 4.4 and Figure 4.5 respectively. The false positives and false negatives count is shown in Figure 4.3.

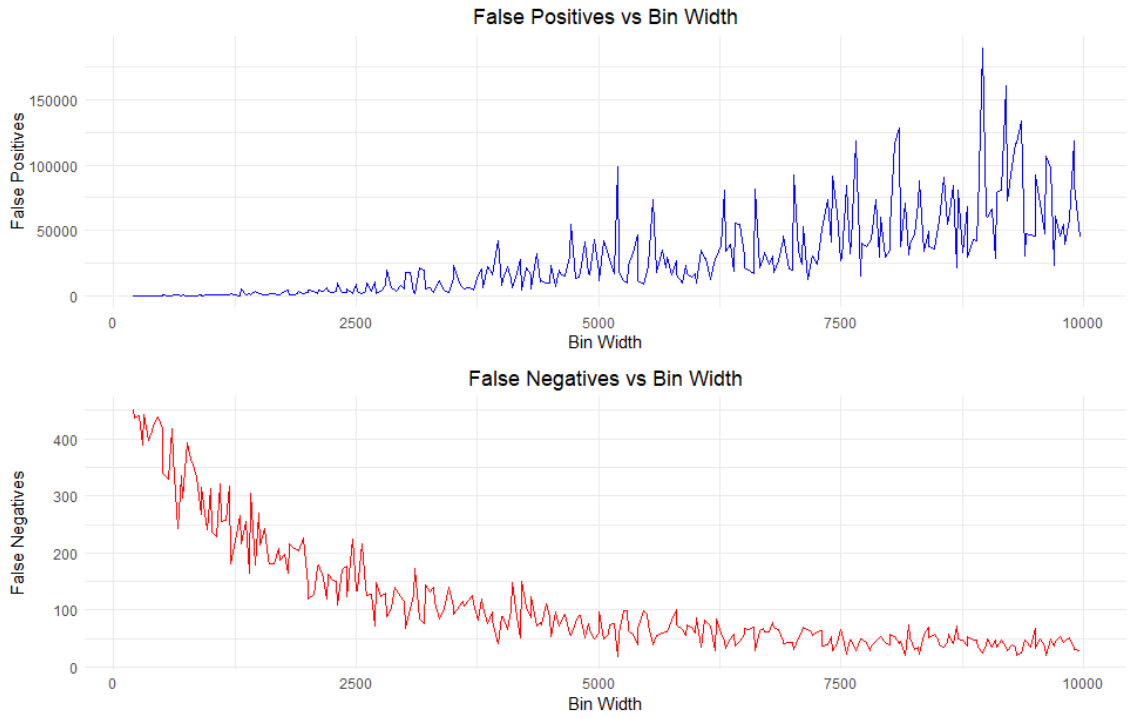

Figure 4.3: The upper graph shows the number of false positives vs. bin width for the testing data and the lower graph shows the number of false negatives vs. bin width for the testing data. There are 1000 points for one party and there are 451 true collision

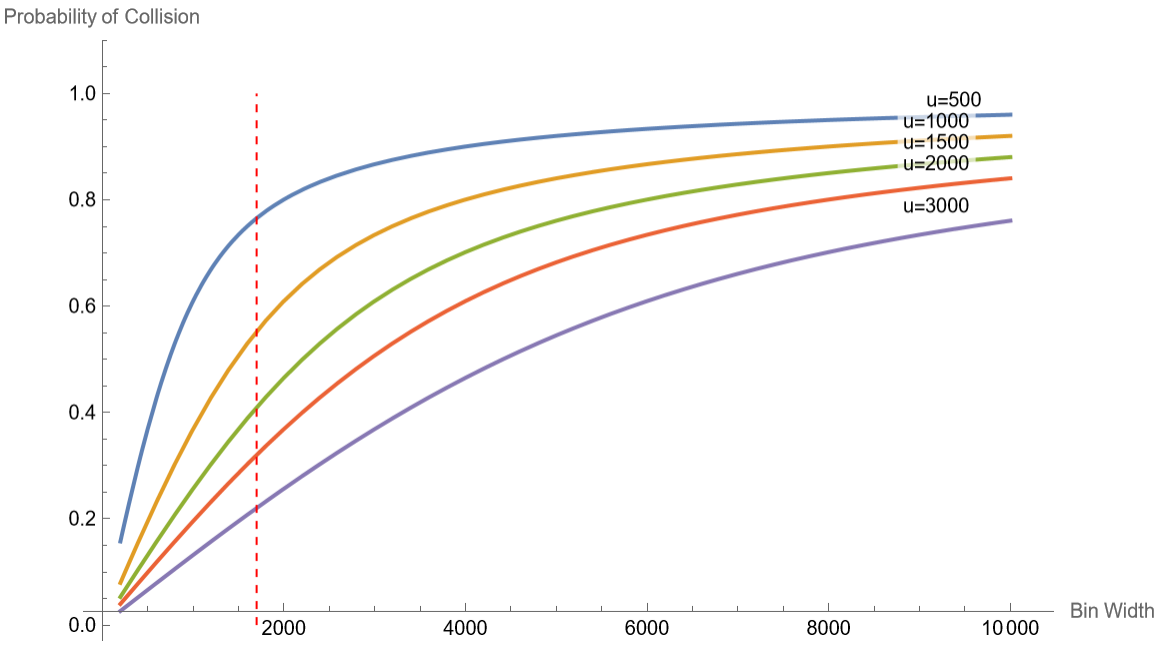

Figure 4.4: The probability of collision calculated by the formula derived in the method section for distance of pairs of 500km, 1000km, 1500km, 2000km, and 3000km.

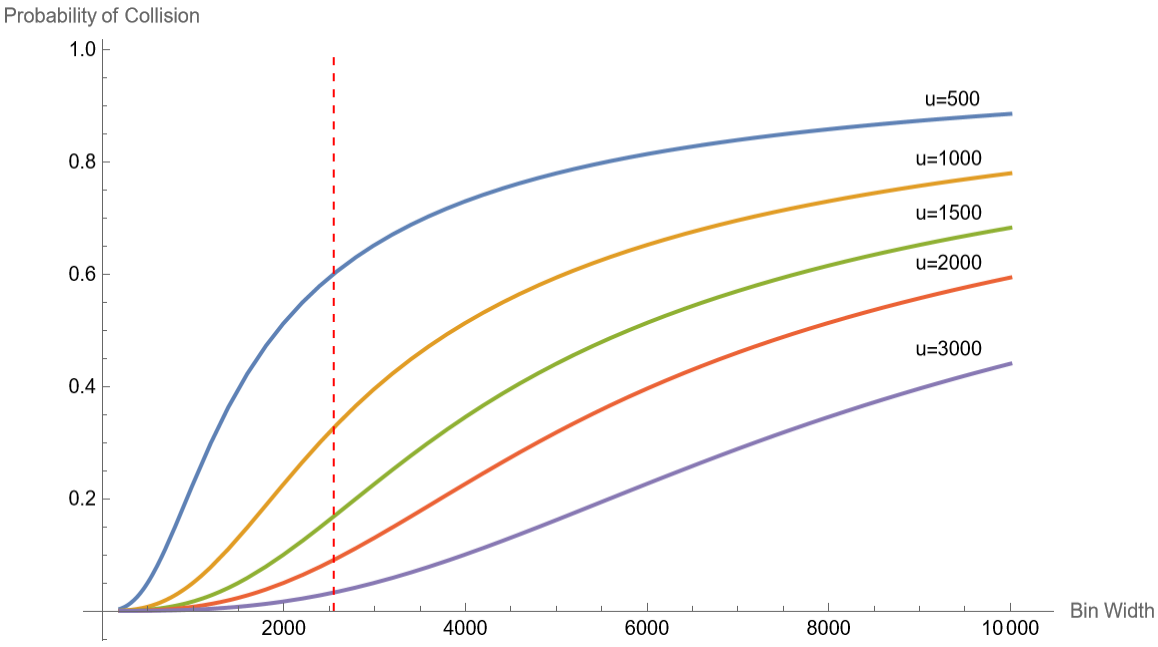

Figure 4.5: The probability of collision calculated by the formula derived in the method section for distance of pairs of 500km, 1000km, 1500km, 2000km, and 3000km for three rounds of hashing.

Chapter 5

Conclusion

As the number of satellites and space debris increases, tracking and predicting their positions has become essential to prevent collisions. The capacity to manage and process this data varies significantly among nations, with only a few possessing advanced monitoring capabilities. These nations not only manage a large volume of satellites but also dominate control over Space Situational Awareness (SSA) data. Current collision avoidance methods primarily rely on central databases like the Space Surveillance Network, which poses risks of data manipulation and often provides less accurate data than what satellite operators can independently track. This underscores the need for improved data-sharing systems among operators. In response, the RAND Corporation proposed a novel approach in 2014 using secure Multiparty Computation (MPC) that allows operators to calculate collision probabilities without sharing private trajectory data, enhancing privacy and reducing reliance on unreliable central databases. However, this method is computationally intensive and may not be practical for operators managing multiple satellites.

This thesis introduces two methods for computing colliding satellites. The first method, employing fuzzy Private Set Intersection (PSI), which requires approximately 75 million AND gates when each of two parties owns 500 satellites. The second method combines Locality-Sensitive Hashing (LSH) with PSI and uses about 25 million AND gates for the same scenario. The second method sacrifices some accuracy to achieve faster processing times, depending on the parameters chosen.

The main contribution of these methods is that they eliminate the need for conjunction analysis calculations, which involve the computation of complex integrals, exponential functions, matrix operations, and multiple arithmetic. Rather than calculating the probability of collision, these methods directly identify potential satellites at risk of collision based on the provided collision distance parameter.

However, this proposed MPC-based solution faces its own set of challenges in practical implementation. Firstly, since a publicly accessible database with space objects' information already exists, it is somewhat unrealistic to assume that parties will not consult this resource or factor its information into their calculations. Thus, one future direction is to incorporate the central database into the paradigm to use as a cross-reference, either to increase accuracy or to decrease running time. Another direction is to find optimal parameters (bin width, collision distance) mathematically or find another LSH function with less false positives and false negatives.

Appendix A

p-stable distribution

De nition A.0.1 (p-stable distributions)**.** A distribution D is p-stable if, for any independent identically distributed (iid) random variables $X_1, \ldots, X_n$ distributed according to D, and any real numbers $v_1, \ldots, v_n$, the random variable $\sum_{i=1}^{n} v_i X_i$ has a probability distribution that is the same as that of the random variable

$$\left(\sum_{i=1}^{n} |v_i|^p\right)^{\overline{p}} X,$$

where X is drawn from D.

Remark. The standard Cauchy distribution is 1-stable and the standard Gaussian probability distribution is 2-stable.

Recall that the characteristic function,

$$\varphi_X(t) = \quad e^{itX}\big],$$

is a function that completely determines the behavior and properties of the probability distribution of the random variable X.

Stable distributions can be parameterized by four parameters, $\quad, \beta, \gamma, \delta$. These parameters can be interpreted as follows:

is the weight in the tails of the distribution. In other words, they are the values outside of the critical values in the distribution.

β is the skewness of the distribution and $\quad 1 < \beta < 1$. A zero beta means that the distribution is symmetric.

γ measures the dispersion of the distribution.

δ is the location parameter.

A normal distribution with mean μ and variance σ is 2-stable and parametrizable by $(2, 0, \sqrt{\sigma}, \mu)$.

Appendix B

Conjunction Analysis

Conjunction analysis calculates the probability of collision for two satellites. Each satellite is modelled as a spherical object, so its radius captures its shape. Each satellite is assumed to deviate from its position, p, and these deviations are assumed to be normally distributed with covariance matrix C. The two satellites are also assumed to have linear relative velocities as in any short time window, the satellite's trajectory is almost linear. Because the positional errors on the two satellites are assumed to be independent and what matters for collision are the relative distance, we can shift all the errors onto one body. It is also standard to shift all the mass onto the other body, creating a "combined object" whose radius is equal to the sum of the radii of the two individual spheres. We can then imagine a ball of radius $R_a + R_b$ passing through a density ellipsoid with covariance matrix $C_a + C_b$. This ball traces a "collision tube" through the combined density ellipsoid, and the probability of collision is then simply the probability mass of the density ellipsoid within this collision tube. The full conjunction analysis calculation is described in Algorithm 16.

Algorithm 3 The conjunction analysis calculation

Inputs: $\{V_i, C_i, P_i, R_i\}_{a,b}$
$V_r \leftarrow V_b \quad V_a$
$i \leftarrow \frac{V_r}{|V_r|}, j \leftarrow \frac{V_b \times V}{|V_b \times V\ |}, k \leftarrow i \times j$
$Q \leftarrow [j\ k]$
$C \leftarrow Q^T(C_a + C_b)Q$
$(u, v) \leftarrow$ Eigenvectors(C)
$(\sigma_x^2, \sigma_y^2) \leftarrow$ Eigenvalues(C)
$\sigma_x \leftarrow \sqrt{\sigma_x^2}, \sigma_y \leftarrow \sqrt{\sigma_y^2}$
$u \leftarrow \frac{u}{|u|}, v \leftarrow \frac{v}{|v|}$
$U \leftarrow [u\ v]$
$\begin{bmatrix} x_m \\ y_m \end{bmatrix} \leftarrow U^T Q^T (P_b \quad P_a)$

$p \leftarrow \frac{1}{2\ \sigma_x \sigma_y} \int_{R}^{R} \int_{\sqrt{R^2\ \ x^2}}^{\sqrt{R^2\ \ x^2}} f(x, y)\ dydx$
Where
$f(x, y) = \exp\left(\quad \frac{1}{2} \left[\left(\frac{x\ \ x_m}{\sigma_x} \right)^2 + \left(\frac{y\ \ y_m}{\sigma_y} \right)^2 \right] \right)$
Return: p

References

[1] T. S. Kelso, D. A. Vallado, J. Chan, B. Buckwalter *et al.*, "Improved conjunction analysis via collaborative space situational awareness," in *9th Advanced Maui Optical and Space Surveillance Technologies Conference, Maui, HI*, 2008.

[2] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982, pp. 160–164.

[3] B. Hemenway, W. Welser, and D. Baiocchi, *Achieving higher- delity conjunction analyses using cryptography to improve information sharing*. Rand Corporation, 2014.

[4] B. Hemenway, S. Lu, R. Ostrovsky, and W. Welser Iv, "High-Precision Secure Computation of Satellite Collision Probabilities," in *Security and Cryptography for Networks*, V. Zikas and R. De Prisco, Eds. Cham: Springer International Publishing, 2016, vol. 9841, pp. 169–187, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-44618-9_9

[5] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 1–19.

[6] D. Morales, I. Agudo, and J. Lopez, "Private set intersection: A systematic literature review," *Computer Science Review*, vol. 49, p. 100567, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013723000345

[7] X. Carpent, S. Faber, T. Sander, and G. Tsudik, "Private set projections & variants," in *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, 2017, pp. 87–98.

[8] K. Nomura, Y. Shiraishi, M. Mohri, and M. Morii, "Secure association rule mining on vertically partitioned data using private-set intersection," *IEEE Access*, vol. 8, pp. 144 458–144 467, 2020.

[9] P. Rindal and M. Rosulek, "Malicious-secure private set intersection via dual execution," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1229–1242.

[10] T. Jiang and X. Yuan, "Traceable private set intersection in cloud computing," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2019, pp. 1–7.

[11] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *NDSS*, 2012.

[12] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure {Two-Party} computation using garbled circuits," in *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[13] L. Malka, "Vmcrypt: modular software architecture for scalable secure computation," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 715–724.

[14] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella *et al.*, "Fairplay-secure two-party computation system." in *USENIX security symposium*, vol. 4. San Diego, CA, USA, 2004, p. 9.

[15] E. Angel, *Interactive Computer Graphics : A Top-Down Approach with OpenGL*. Boston, MA: Addison Wesley Longman, 2000.

[16] G. Campiti, G. Brunetti, V. Braun, E. Di Sciascio, and C. Ciminelli, "Orbital kinematics of conjuncting objects in Low-Earth Orbit and opportunities for autonomous observations," *Acta Astronautica*, vol. 208, pp. 355–366, Jul. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576523002060