Privacy-Preserving Drone Navigation Through Homomorphic Encryption for Collision Avoidance

Allan Luedeman
Dept. of Electrical & Computer Engineering
Florida International University
Miami, FL, USA
alued004@fiu.edu

Nicholas Baum
Dept. of Electrical & Computer Engineering
Florida International University
Miami, FL, USA
nbaum005@fiu.edu

Andrew Quijano*
Dept. of Computer Science & Engineering
New York University
New York, NY
andrew.quijano@nyu.edu

Kemal Akkaya
KF School of Computing and Information Sciences
Florida International University
Miami, FL, USA
kakkaya@fiu.edu

***Abstract*—As drones increasingly deliver packages in neighborhoods, concerns about collisions arise. One solution is to share flight paths within a specific zip code, but this compromises business privacy by revealing delivery routes. For example, it could disclose which stores send packages to certain addresses. To avoid exposing path information, we propose using homomorphic encryption based comparison to compute path intersections. This allows drones to identify potential collisions without revealing path and destination details, allowing them to adjust altitude to avoid crashes. We implemented and tested our approach on resource-limited virtual machines to mimic the computational power of drones. Our results demonstrate that our method is significantly faster and requires less network communication compared to a garbled circuit-based approach. We also provide a security analysis of the approach against potential attacks.**

***Index Terms*—drone privacy, homomorphic encryption, drone navigation**

I. Introduction

Drone technology is rapidly proliferating in various domains, such as aerial surveillance [4], package delivery [2], and search and rescue operations [3], [14], with many industries seeing significant advancements. However, data privacy concerns arise as drones are responsible for more services, with one particular instance being the usage of delivery drones. Delivery drones are cost-effective because companies can reduce the costs associated with delivery drivers, fuel, and vehicle maintenance. Drones can also bypass ground traffic and take more direct routes, significantly reducing delivery times compared to traditional delivery methods. Finally, drones can operate outside typical delivery hours, offering faster and more flexible delivery options for consumers.

However, as more delivery drones are being used, collision avoidance becomes a serious concern. Traditional collision avoidance mechanisms often involve sharing detailed flight paths between drones, raising significant privacy concerns. This information may include specific home addresses, shopping patterns, and other data related to individual privacy. If the data is compromised, criminals might target specific homes or businesses for theft or other malicious activities, as they know when valuable packages are expected to arrive. Also, it is important to keep this information private between drones owned by competitors, as leaked delivery data could be used to gain insight into a company's supply chain, customer base, and operational practices, leading to potential business espionage.

To address the need for privacy, various approaches are used, such as homomorphic encryption, secure multiparty computation (MPC) [8], or garbled circuits [12]. Homomorphic encryption techniques enable parties to perform computations on encrypted data without revealing the underlying information. A garbled circuit is a cryptographic protocol that is used to securely evaluate a function over encrypted inputs without revealing it to the parties involved. It involves converting a Boolean circuit into an encrypted form where the input, output, and internal wires are encoded in such a way that only the intended computation can be performed without leaking any additional information. Li et al. [12] used garbled circuits to compute whether a collision will occur with two robots. If a collision is detected, both entities would randomize their paths to avoid the collision. The MPC based approach requires a shared grid in which all participating drones must operate, and the processing time escalates with increased size or resolution of the grid, creating scalability issues. Homomorphic encryption approaches, on the other hand, can be customized to the needs of the application without needing extra pre-computations. Depending on the arithmetic operations required, the most efficient homomor-

*This paper was completed outside of his job responsibilities at Amazon. The views expressed in this paper are those of the author and do not necessarily reflect the official policy or position of Amazon.

phic implementations can be chosen.

In this paper, using homomorphic encryption, we present a privacy-preserving drone collision avoidance protocol based on the line intersection algorithm [9], which is faster than existing collision avoidance protocols [8] [12] and uses less network bandwidth. Our protocol assumes that two drones that have different owners are about to collide. To avoid over-complicating paths, we assume all paths are in a 2-dimensional plane. We utilize an intersection algorithm that operates on encrypted paths using homomorphic encryption, enabling drones to detect potential collisions without revealing the full path of the other drone. The drones would start this protocol once they are within a range that is reasonable enough to cause a collision. If a collision is going to occur, a drone will temporarily change its altitude (that is, modifying the Z coordinate) and return to its default altitude once its path no longer overlaps with the other drone. We are the first to offer the encrypted version of the path comparison algorithm without relying on any pre-computations.

Our approach has other advantages compared to related work based on our protocol. First, it is an efficient collision avoidance solution that avoids re-computing a path that could cause delays [12]. Indeed, evaluation results confirm that we can gain about 30% latency reduction. Also, unlike [8], since we assume we are on a 2-dimensional plane, we can use GPS coordinates as our path, which has the advantage of not requiring any pre-computed matrices for our protocol to work.

The remainder of the paper is organized as follows. In the next section, we provide background information in both homomorphic encryption algorithms and an intersection decision algorithm. Then, Section III discusses related work in the domain of secure route computation. In Section IV, we establish our assumptions, system design, and threat model. Section V discusses our experiment results. Finally, Section VI concludes the paper.

II. Related Work

Privacy-preserving collision avoidance has started to receive some recent attention in various contexts. For example, Li et al. [12] developed a privacy-preserving multi-robot planning protocol that would ensure two robots on a floor would not collide. A garbled circuit is a function represented by a Boolean circuit with logic gates. Using a two-party MPC approach, one party is assigned the role of *garbler* which generates the garbled circuit, while the other party is called the *evaluator*. The inputs of each party are then encrypted, and the *evaluator* processes the garbled gates with the garbled (encrypted) inputs to compute the garbled output. The *evaluator* can then map this garbled output back to the actual output values using the mapping provided by the *garbler* [24]. In Li et al.'s implementation, the output is a Boolean decision on whether an intersection is present somewhere in the complete path. It does not provide exact intersection locations.

Sciancalepore et al. [19] introduce Privacy-Preserving Trajectory Matching (PPTM), a protocol that enables UAVs to discover potential spatial and temporal collisions without sharing sensitive location and timestamp data with untrusted entities. PPTM employs a tree-based algorithm called Incremental Capsule Matching and integrates a lightweight privacy-preserving proximity testing solution for private comparisons. This approach improves path planning efficiency, reduces delivery time, and minimizes energy consumption for UAVs. However, it does not always guarantee accurate solutions. It trades accuracy and performance.

Finally, Desai et al. [8] propose a secure MPC for trajectory planning among drones, ensuring collision detection without disclosing sensitive information. It uses matrix representation and matrix addition as an approach to compare trajectories, thus reducing computational complexity and improving performance. The matrix representation involves considering each path to take place on a discrete grid of the region. Each drone's path is represented by a series of continuous cells of value one, where anyplace the drone will not pass through is given a value zero. The matrix must be consistent between the drones, and so must cover the entire area of possible flight, and cannot handle traffic that is not familiar with the local matrix, such as a drone from another area.

Compared to the approach of Li et al. [12], our approach is faster and lighter on network traffic (as will be shown in Section V), as well as allowing for planning by more than two drones by allowing for deterministic path changes in the event of an intersection rather than random movement. Then, Sciancalepore et al. [19] the most lightweight implementation of their protocol does have false positives and may not be trustworthy in applications where safety is critical such as drone delivery or military surveillance. Finally, compared to Desai et al. [8], our approach eliminates the need for a discrete matrix of paths possibilities, instead taking GPS coordinates as input. This allows for more detailed planning and also bases computation time solely on the complexity of the path, and not on the size of the delivery area being considered. In addition, a drone coming into an area from a far distance could utilize our method because it does not need to have a common matrix of possible locations.

III. Background

A. Homomorphic Encryption

An Additively Homomorphic Encryption scheme such as Paillier [16] or DGK [5] [6] [7] allows a user to combine two ciphertexts to receive the sum of both ciphertexts. Assume there are two messages α, $\alpha^{'}$ in the plaintext space $\mathbb{M}$ and we define the encryption of α as $[\![\alpha]\!]$. We use the 'boxplus' operator ($\boxplus$) to denote the addition of two ciphertexts such that upon decryption $[\![(\alpha + \alpha^{'})]\!] = [\![\alpha]\!] \boxplus [\![\alpha^{'}]\!]$.

In Paillier [16], we define m as the plaintext message, n is the product of primes p and q, r is a random number and g is a parameter of the encryption scheme, as shown in Eq. 1.

$$[\![m]\!] = g^m r^n \mod n^2 \tag{1}$$

If you multiply two Paillier ciphertexts, the output would be an encrypted addition of the plain-text. A Paillier ciphertext that is exponentiated with a plaintext value would return the encrypted product of the ciphertext and plaintext as in Eq. 2.

$$\begin{aligned} [\![\alpha + \alpha']\!] &= (g^{\alpha} r^n)(g^{\alpha'} r^n) = g^{\alpha+\alpha'} r^{2n} \\ [\![(\alpha)(\alpha')]\!] &= (g^{\alpha} r^n)^{\alpha'} = g^{(\alpha)(\alpha')} r^{(\alpha')(n)} \end{aligned} \tag{2}$$

However, Paillier does not have an operation where you can obtain the product of two ciphertexts. However, using a two-party protocol [23] that takes advantage of the distributive property of multiplication, we can obtain the product of two Paillier [16] ciphertexts. In this protocol, we assume that Alice has $[\![x]\!]$ and $[\![y]\!]$, and generates two random values a and b to additively blind each encrypted value, respectively. Alice sends $[\![(x+a)]\!]$ and $[\![(y+b)]\!]$ to Bob to decrypt, multiply, and return the product. Then Alice could use the following operations to obtain $[\![xy]\!]$, as shown in Eq. 3.

$$[\![xy]\!] = [\![(x+a)(y+b)]\!] \boxplus [\![-bx]\!] \boxplus [\![-ay]\!] \boxplus [\![-ab]\!] \tag{3}$$

Finally, another application of homomorphic encryption is to compare it to encrypted numbers. Assume that there are two t-bit integers held by party A and party B, which are represented in binary form, $x = \sum_{i=0}^{t-1} x_i 2^i$ and $y = \sum_{i=0}^{t-1} y_i 2^i$. The objective of parties A and B, respectively, is to obtain the protocol bits δ_A and δ_B, such that $\delta_A \oplus \delta_B = \mathbb{1}\{x \leq y\}$.

The secure integer comparison problem was first stated in [26], where two millionaires would like to determine who is richer without revealing the amount of their wealth. The first solution to the problem was introduced in [13] and various more advanced approaches such as the implementation of the DGK cryptography system and its comparison protocol [5] [6] [7]. The high communication cost of these comparison protocols was addressed [21] [22] and finally [11] improved them to be resistant to timing attacks.

B. *Encryption Library*

We utilize a Java library implemented by Quijano and Akkaya [18] [17], which implements the encrypted equality check [15], multiplication over two homomorphic ciphertexts [23] and uses the Joye and Salehi encrypted integer comparison protocol [11]. The encrypted integer, multiplication, and equality check can be used for pairs of Paillier [16] ciphertexts or DGK ciphertexts [5] [6] [7]. We use these functions to implement a privacy-preserving version of a line intersection algorithm [9].

C. *Threat Model*

We assume Alice and Bob, representing two drones, respectively, to be honest but curious, so we expect both parties to faithfully follow the encrypted intersection algorithm, but will attempt to gather as much information as possible from each other such as path information, destination location, etc. We also assume that there may be external parties that will attempt to obtain the path information of the drones.

As an attack, we consider that Alice may attempt to replicate Bobs' path even if it is encrypted through intersections by defining her own path as a set of very small line segments that cover an entire area to a resolution that will give her Bob's path information, as well as a close replica of a complete path.

IV. Proposed Approach

A. *System Model and Scenario Overview*

In our scenario, we assume the two drones that are engaging with each other through our protocol to be owned by different companies, e.g., Amazon and UPS. In this scenario, Alice is a drone about to take off, and Bob is either in flight or also ready for take-off. We consider a specific timeframe where Alice can send a request for paths of any other drones that are or will be flying in a neighborhood along with its speed. This would require communication between drones at longer distances than local area networks (e.g. WiFi-like ranges). This can be achieved by employing protocols such as 5G NB IoT [20] or LoRa [25]. This neighborhood should be much larger than the area covered during the flight to cover all drones that could come into the flight area during flight time. Bob, which is or will be flying in the area Alice is broadcasting to, responds. Note that we expect these drones to have different starting and delivery locations. Thus, it would be unlikely that the same pair of drones would possibly collide multiple times in an honest path. However, we assume that all drones have a default altitude enforced by regulations

B. *Encrypted Comparison of Paths*

We rely on homomorphic encryption to be able to compare the paths of two drones without exposing any information to each other regarding the path including any intersection point. In this approach, Bob has a key pair and a route composed of an arbitrary number of line segments. Alice also has a route composed of an arbitrary number of line segments. A sample route along with their line segments is shown in Fig. 1.

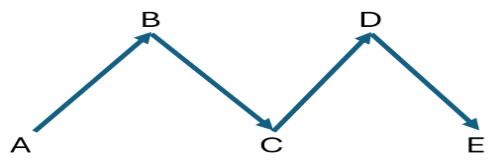

Fig. 1: A Path is compromised of an arbitrary number of lines segments, e.g. (A, B), (B, C), (C, D), (D, E)

Both Alice and Bob will follow the steps below to run our approach, as also shown in Fig. 2:

1) Bob will encrypt his route using his own public key and await a connection from Alice.
2) Upon connection to Alice, Bob sends his public key and encrypted route to Alice.
3) Alice then uses Bobs public key to encrypt her route, and they both complete an encrypted version of the route using Algorithm 2.

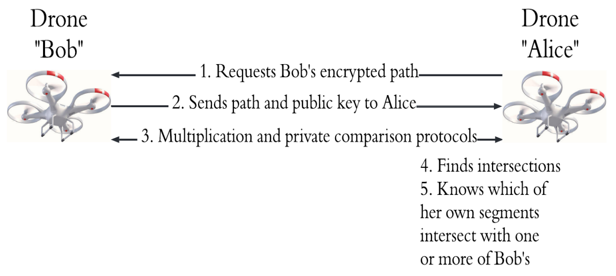
Drone "Bob"
Drone "Alice"
1. Requests Bob's encrypted path
2. Sends path and public key to Alice
3. Multiplication and private comparison protocols
4. Finds intersections
5. Knows which of her own segments intersect with one or more of Bob's

Fig. 2: Protocol for two drones to determine which intersections there might be collisions to avoid.

Once the algorithm ends, Alice knows which of her line segments on her path will collide with Bob. As Bob's path is encrypted, no other information is available to Alice. Bob does not receive the results of the comparisons and, if he wishes to obtain the same information, the protocol needs to be run again with the roles switched. However, this is not necessary, as it would be assumed that Alice would change altitude to avoid colliding with Bob.

Algorithm 1 Drone behavior in flight

```
    Input: Protocol-Initiation-Range

 1: while In Flight do
 2:     if Another drone is in Protocol-Initiation-Range then
 3:         Determine which Drone is Alice and Bob
 4:         Segments ← Encrypted version of Algorithm 2
 5:         for Segment in Path do
 6:             if Collision occurs at Segment i then
 7:                 Alice adjusts altitude
 8:             else
 9:                 Return to default altitude
10:             end if
11:         end for
12:     else
13:         return continue
14:     end if
15: end while
```

C. Computing Intersections

The line segment intersection algorithm [9] (that is, Algorithm 2) we use is based on the concept of orientation of an ordered triplet. There are three possible orientations for any ordered triplet of points: *clockwise*, *counterclockwise*, or *collinear*. To demonstrate this, we used the points (A, B, C), shown in their three possible orientations in Figure 3.

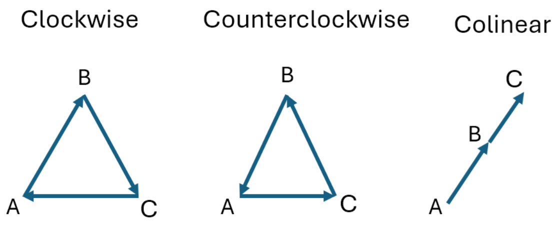

Fig. 3: Three cases of orientation for points A, B, and C.

Assume that we have two line segments (A , B) and (C, D) that intersect, there are two cases that would occur.

In the general case, where two line segments intersect, (A, B, C) and (A, B, D) have different orientations and the orientations of (C, D, A) and (C, D, B) are also different. In the other case, if all line segments are collinear, they intersect if their x-projection and y-projections overlap. The pseudocode to find the orientation of an ordered triplet (A,B,C) composed of points (A_x,A_y), (B_x,B_y), and (C_x,C_y), where the subscript identifies the plane of projection.

To handle the case that two segments are collinear, we check if any of the four points is located in the other segment.

We can compose a routine that will decide whether two line segments (A, B) and (C, D) intersect. The pseudocode for this is provided in Algorithm 2.

Algorithm 2 Line Segment Intersection Algorithm [9]

```
    O1 = Orientation(A,B,C)
    O2 = Orientation(A,B,D)
 3: O3 = Orientation(C,D,A)
    O4 = Orientation(C,D,B)
    if O1 $\neq$ 02 $\wedge$ O3 $\neq$ 04 then
 6:     return True
    end if
    if O1, O2, O3, O4 are all Collinear then
 9:     if    A is on segment (C,D) $\vee$
        B is on segment (C,D) $\vee$
        C is on segment (A,B) $\vee$
12:     D is on segment (A,B) $\vee$ then
            return True
        else
15:         return False
        end if
    end if
```

D. Security Considerations

Since the paths are encrypted, neither Alice nor Bob and no other third party can decrypt any information collected from the wireless broadcasts.

Now, let us assume Alice will be logging the results of the collision for each line segment in her path with Bob in an attempt to reverse engineer Bob's path. If there is no collision, privacy is preserved. If there is only one collision, Alice would know that one line segment would intersect with Bob. Since there is no time information, Alice would not be able to determine where in the line segment Bob would collide, or Bob's speed or direction.

During the engagement with Bob, should Alice decide to run the intersection protocol repeatedly against a series of closely spaced parallel lines composed of short segments, as shown in Figure 4, she could compose an approximation of Bob's path through the intersections she discovers.

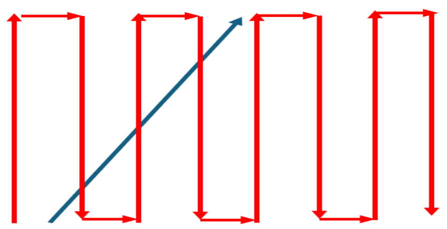

Fig. 4: The red path is the path of an Alice drone attempting to brute force, the blue path of a Bob drone.

The first mitigation of this attack is that, based on our assumptions in Section III-C, the drones would have different starting points, so realistically it would still be difficult for Alice to randomly find Bob to attempt this sort of path. This attack also does not factor in that an Alice drone attempting this brute-force approach could just as easily miss fast-moving Bob drones [10] that came too early or too late within the long parallel line segments. This attack requires Alice and Bob to be within range of each other and to talk to each other for a very long time, which is not possible. Finally, drones also have a very limited range [1], so if Alice wanted to complete both her delivery and try this task, it would be risky, as the drone increases its risk of running out of energy before returning to its refueling station.

V. Performance Results

A. Experiment Setup

We implemented our protocol entirely in Java and used a homomorphic encryption library implemented in Java [18] [17]. We open-sourced the code we used to implement our collision avoidance algorithm[1].

We considered the following metrics to access performance:

- *Execution Time*: This is the time for computing if an intersection would occur between two line segments
- *Network Traffic*: We count the number of bytes that both Alice and Bob write to a socket during the protocols.

Thirty random x-y plane line segments with values from -99 to 99 were generated because [12] does not provide intersection locations. To fairly compare speed, we used single segments, so both protocols returned the same result: a boolean decision on the existence of an intersection. The experiment was completed on a pair of Debian 12 virtual machines with 2 CPUs and 4 GB of RAM, which were chosen to emulate the resources of a Raspberry Pi 4[2]. We calculated the average time and network traffic size for each comparison based on these 30 trials.

B. Experimental Results

1) Comparison with Li et al.: When comparing our work with Li et al. [12], we both used 2048-bit keys. The results in Table I indicate that our approach is 30% faster and sends less data over the network compared to Li et al. [12]. This is because of the efficiency of the comparison operation in DGK. In contrast, MPC-based approaches require a lot of communications and computation to perform comparison operations. The dominating nature of comparison operations in path comparison shows that any time savings should focus on this operation, and use of our approach supports this observation. The gains will even improve with an increase in the number of segments for a given path comparison since only a single segment is focused on average.

TABLE I: Comparing the computation time and bytes sent over a network with [12].

Approach	**Time (seconds)**	**Network traffic (bytes)**
Our Approach	4.407s	4634
Li et al. [12]	6.092s	39221

The improvements in speed and network traffic means that our approach would be scalable for multiple pairs of drones to compare their paths within a certain airspace. Our protocol would have the edge if used in networks where bandwidth is limited, such as using LoRa.

2) Comparison with Desai et al.: We also performed an experiment to compare our approach with Desai et al. We used two Raspberry Pi 4s to have a fair comparison (as done in that work). We obtained an average run-time of 16.531 seconds, which is faster than Dessai et al. [8] taking 30 seconds for a matrix of size 96 and using a secret share size of 64 bits.

It is important to understand that the speed of our approach and that of Dessai et al. [8] vary based on different parameters. Route complexity does not change the speed of Dessai et al.'s [8] approach as it does ours, but the area and resolution considered for route comparisons does. But a matrix size of 96 is very limiting when we consider realistic drone flight areas, and Desai et al. [8] computation time scales exponentially with increased matrix size.

VI. Conclusion and Future Work

In this paper, we implemented a new encrypted intersection algorithm using Homomorphic encryption, which can be used

[1]https://github.com/adwise-fiu/homomorphic-path-comparison

[2]https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

by two drones owned by separate entities to compute where they would intersection. Our algorithm is more flexible than [8] as we can use GPS coordinates as part of our line segments, meaning more flexibility on where the protocol could work, as well as not requiring all participating drones to share a finite possible flight area. Furthermore, the results of the experiment showed that our algorithm outperforms the existing approaches. It is faster than both Li et al.'s [12] collision protocol and Desai et al. MPC protocol and requires less network bandwidth.

The primary privacy concern, a brute force attack designed to reconstruct a flight path out of detected collision points, would take far longer than the flying drone would be in the air, and would fail due to lost connectivity between the drones. The large difference in computation time between an honest path and a brute-force attack also leaves open the possibility of stopping the attack based on assumptions about time and resources used in an honest implementation.

The aforementioned use of GPS coordinates as input allows for the potential for our approach to be used in-flight in real-time, as opposed to route planning. Using this approach in this way is currently limited by computation time, but as drones increase in computation power and wireless networks increase in bandwidth, it is possible that this approach be used in an in-flight scenario.

Acknowledgements

This work was supported by the Griffiss Institute (Award Number SA10012022020481). The content of this paper is solely the responsibility of the authors and does not necessarily represent the official views of the sponsor. Allan Luedeman and Nicholas Baum would like to thank the Virtual Institute of Cyber Operations and Research (VICOR) for their funding and support. Finally, we thank Alfredo Bayuelo for his assistance in comparing our work with the work of Li et al. [12]. We thank Alfredo Bayuelo for his assistance in comparing our work with the work of Li et al. [12].

References

[1] Sherilyn Beall. What is the range on a drone. https://robots.net/tech/what-is-the-range-on-a-drone/. [Online; accessed 23-May-2024, Published; 19-October-2023, Modified; 28-February-2024].

[2] Fabio Borghetti, Claudia Caballini, Angela Carboni, Gaia Grossato, Roberto Maja, and Benedetto Barabino. The use of drones for last-mile delivery: A numerical case study in milan, italy. *Sustainability*, 14(3), 2022.

[3] Tiziana Calamoneri, Federico Corò, and Simona Mancini. A realistic model to support rescue operations after an earthquake via uavs. *IEEE Access*, 10:6109–6125, 2022.

[4] Gabriel Carrasco-Escobar, Marta Moreno, Kimberly Fornace, Manuela Herrera-Varela, Edgar Manrique, and Jan E. Conn. The use of drones for mosquito surveillance and control. *Parasites and Vectors*, 15(1):473, 2022.

[5] Ivan Damgaard, Martin Geisler, and Mikkel Kroigaard. Efficient and secure comparison for on-line auctions. In *Australasian conference on information security and privacy*, pages 416–430. Springer, 2007.

[6] Ivan Damgaard, Martin Geisler, and Mikkel Kroigaard. Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography*, 1(1):22–31, 2008.

[7] Ivan Damgaard, Martin Geisler, and Mikkel Kroigard. A correction to 'efficient and secure comparison for on-line auctions'. *International Journal of Applied Cryptography*, 1(4):323–324, 2009.

[8] Anushka Desai, Oscar G. Bautista, and Kemal Akkaya. Privacy-preserving collision detection for drone-based aerial package delivery using secure multi-party computation. In *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, MobiHoc '23, page 510–515, New York, NY, USA, 2023. Association for Computing Machinery.

[9] Cormen Thomas H. *Introduction to Algorithms*. PHI Learning Pvt. Ltd., 2009.

[10] La Verne Haun. How fast can a delivery drone fly. https://robots.net/tech/how-fast-can-a-delivery-drone-fly/. [Online; accessed 23-May-2024, Published; 20-October-2023, Modified; 02-March-2024].

[11] Marc Joye and Fariborz Salehi. Private yet efficient decision tree evaluation. In *Proceedings of DBSec 2018*, pages 243–259, 07 2018.

[12] Li Li, Alfredo Bayuelo, Leonardo Bobadilla, Tauhidul Alam, and Dylan A. Shell. Coordinated multi-robot planning while preserving individual privacy. In *International Conference on Robotics and Automation (ICRA)*, pages 1–7, 2018.

[13] Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *International Conference on Applied Cryptography and Network Security, ACNS 2005*, pages 456–466, 06 2005.

[14] Ignacio Martinez-Alpiste, Gelayol Golcarenarenji, Qi Wang, and Jose Maria Alcaraz-Calero. Search and rescue operation using uavs: A case study. *Expert Systems with Applications*, 178:114937, 2021.

[15] Majid Nateghizad, Thijs Veugen, Zekeriya Erkin, and Reginald L Lagendijk. Secure equality testing protocols in the two-party setting. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–10, 2018.

[16] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 223–238. Springer, 1999.

[17] Andrew Quijano. Homomorphic encryption library. https://github.com/adwise-fiu/Homomorphic_Encryption, 2019.

[18] Andrew Quijano and Kemal Akkaya. Server-side fingerprint-based indoor localization using encrypted sorting. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, pages 53–57, 2019.

[19] Savio Sciancalepore and Dominik Roy George. Privacy-preserving trajectory matching on autonomous unmanned aerial vehicles. In *Proceedings of the 38th Annual Computer Security Applications Conference*, ACSAC '22, page 1–12, New York, NY, USA, 2022. Association for Computing Machinery.

[20] Hanif Ullah, Nithya Gopalakrishnan Nair, Adrian Moore, Chris Nugent, Paul Muschamp, and Maria Cuevas. 5g communication: An overview of vehicle-to-everything, drones, and healthcare use-cases. *IEEE Access*, 7:37251–37268, 2019.

[21] Thijs Veugen. Improving the dgk comparison protocol. In *WIFS 2012 - Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security*, pages 49–54, 12 2012.

[22] Thijs Veugen. Correction to "improving the dgk comparison protocol". *Cryptology ePrint Archive*, 2018.

[23] Baptiste Vinh Mau and Koji Nuida. Correction of a secure comparison protocol for encrypted integers in ieee wifs 2012 (short paper). In *Advances in Information and Computer Security: 12th International Workshop on Security, IWSEC 2017, Hiroshima, Japan, August 30–September 1, 2017, Proceedings 12*, pages 181–191. Springer, 2017.

[24] Sophia Yakoubov. A gentle introduction to yao's garbled circuits. *preprint on webpage*, 2017.

[25] Evşen Yanmaz, Saeed Yahyanejad, Bernhard Rinner, Hermann Hellwagner, and Christian Bettstetter. Drone networks: Communications, coordination, and sensing. *Ad Hoc Networks*, 68:1–15, 2018.

[26] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164. IEEE, 1982.