MPCTM: MPC-Based Trajectory Matching on Unmanned Autonomous Systems

Dominik Roy George*, Martin Zbudila* and Barry Van Leeuwen
COSIC, KU Leuven, Belgium
{firstname.lastname}@kuleuven.be

***Abstract*—Emerging technologies such as Unmanned Autonomous Systems (UASs) are increasingly being deployed across several application domains. Their deployment, however, introduces safety risks, especially when unmanned vehicles operate in shared operational spaces (e.g., air, land, or maritime) and may follow collision-prone paths. This creates the need for solutions that can detect possible collisions among pre-planned trajectories, thereby improving operational safety and reducing risk. At the same time, revealing trajectories in plaintext leaks sensitive information, such as traversal patterns, deployment and infrastructure locations, and potentially private user data. In this paper, we present a privacy-preserving trajectory matching protocol based on Multi-Party Computation (MPC). We introduce an end-to-end secure scheme for collision detection that reveals no information beyond the prescribed output. Our solution models the behaviour of UASs along their paths, accurately capturing collisions that may occur between discretely sampled points. We further propose a lightweight, MPC-friendly model and discuss the trade-off between accuracy and efficiency for the two approaches. We provide a proof-of-concept implementation of our scheme and compare it to the closest state-of-the-art solution. Our evaluation shows that our work *MPCTM*, in the dealer mode, improves the online latency upon previous work by over $2\times$ in our modelled protocol and by $8\times$ in our lightweight protocol. In autonomous mode, our protocol results in a slow-down of $2.5 - 3\times$, while providing a more expressive model with *no false negatives* and fully preserving privacy of the trajectory data.**

I. Introduction

In recent years, interest in UASs has grown significantly because of their socio-economic benefits and expanding industrial applications. Their versatility enables deployment on land [36], [1], on water [24], [16], and in the air [2], [5] for applications such as surveillance, goods delivery, search and rescue, and traffic monitoring. As these capabilities expand, market interest also grows, with the global UAS market projected to reach $992 billion by 2035 [37]. However, wider deployment increases the risk of collisions between UASs, making rapid and accurate collision detection and avoidance essential for safe operation. As UASs are deployed across an expanding range of applications, automated UAS-on-UAS collision prevention is receiving growing attention from operators and manufacturers. We consider UASs operating in two spatial domains over time: (i) two-dimensional environments, including ground robots, ground vehicles outside regulated traffic settings, and unmanned surface vessels and (ii) three-dimensional environments, including unmanned aerial and underwater vehicles. A naive collision-detection approach requires nearby UASs to exchange their planned trajectories in plaintext and evaluate them using a distance-based matching function. A potential conflict is detected when the corresponding spatial regions and temporal intervals jointly satisfy the collision predicate, allowing the involved UASs to adjust their spatiotemporal trajectories before entering the conflicting region. However, exchanging trajectories creates a serious privacy risk because they may contain sensitive operational information. Disclosure to untrusted parties could reveal operational patterns and enable targeted attacks. For example, the routes and timing of delivery UASs may expose predictable stops, low-speed segments, delivery destinations, or storage locations. An attacker could exploit this information to intercept valuable cargo, disrupt operations, or conduct targeted jamming. This creates a fundamental tension between safety and privacy: UASs must compare their planned trajectories to prevent collisions without revealing the sensitive information they contain.

Existing privacy-preserving trajectory-matching approaches, including [44], [42], [35], [19][1], may reveal the approximate direction of UAS movement through their trajectory representations. In sensitive delivery scenarios, this information could expose likely destinations, operational areas, or future movement patterns. Moreover, these approaches do not jointly account for deviations from planned paths, trajectory sampling rates, and UAS velocity, which determine a vehicle's possible spatiotemporal position. Omitting these factors can therefore reduce collision-detection reliability and weaken the safety guarantees required for industrial deployment. At the same time, UASs rely on limited energy sources, such as batteries or fuel, while performing onboard navigation, sensing, communication, and control in addition to their programmed missions. Privacy-preserving collision detection must therefore account for its computational and energy costs. Although state-of-the-art UASs [34], [29] may offer laptop-class computing capabilities, these resources are shared with concurrent mission-critical tasks and may need to support extended operation without recharging or refuelling. Consequently, privacy-preserving trajectory matching should impose minimal computational, memory, and energy overhead. To address these challenges, we introduce *MPCTM*, a privacy-preserving framework compris-

*These authors contributed equally.

[1]See Table I for the full list

ing two independently designed trajectory-matching protocols: Π_{OBRTM} and Π_{PointTM}. Rather than being dimensional variants of the same predicate, they provide distinct motion models and trade-offs between collision-detection precision and secure-computation cost.

Π_{OBRTM} targets vehicles operating in two-dimensional environments, such as ground robots and unmanned surface vessels. It derives an endpoint-constrained region of feasible movement from consecutive trajectory samples, their timestamps, and the vehicle's velocity bound. This region is enclosed by an oriented bounding rectangle, whose intersections are securely evaluated using MPC. The protocol captures possible deviations between samples and accommodates different sampling intervals, providing a detailed motion model at a higher computational cost.

In contrast, Π_{PointTM} provides lightweight collision detection for vehicles operating in three-dimensional environments, such as aerial and underwater vehicles. It uses dynamically derived spatial bounds and squared coordinate differences, avoiding secure square roots, orientation tests, and complex volumetric-intersection predicates. Its coarser model may report additional potential conflicts but substantially reduces the cryptographic workload for resource-constrained vehicles.

Both protocols protect trajectory coordinates, timestamps, and motion parameters, revealing only the prescribed collision output and, when required, the corresponding conflicting indices. Although designed for two parties, they support multi-vehicle and swarm settings through parallel pairwise executions between vehicles whose operational regions may overlap.

Furthermore, we design *MPCTM* to operate in two deployment modes: (i) an *Autonomous Mode*, in which the two UASs independently execute both the preprocessing and online phases of the MPC protocol and (ii) a *Dealer Mode*, in which a non-colluding semi-honest third party performs the computationally intensive preprocessing operations. These deployment modes allow *MPCTM* to accommodate different operational environments and resource constraints while preserving the privacy of the underlying trajectory data.

Contributions: In this paper, we make the following contributions:

- We present *MPCTM*, an MPC suite comprising two independent protocols for privately detecting conflicts between pre-planned UAS trajectories. We formalise their outputs and prove semi-honest security.
- We design Π_{OBRTM}, an endpoint-constrained protocol that models feasible movement along two-dimensional trajectories using segment endpoints, timestamps, and velocity bounds without synchronised sampling.
- We design Π_{PointTM}, a lightweight protocol for three-dimensional trajectories using dynamic spatio-temporal bounds and squared-coordinate comparisons to avoid costly secure geometry, at the cost of false positives.
- We support *Autonomous Mode*, where the vehicles generate preprocessing material, and *Dealer Mode*, where a non-colluding semi-honest party provides it.
- We implement both protocols in HPMPC [22] and MP-SPDZ [28], covering function-dependent and function-independent preprocessing.
- We evaluate accuracy, runtime, communication, memory, and energy on embedded devices using real-world mobility trajectories. On the evaluated dataset, Π_{OBRTM} achieves a 100% F1 score and Π_{PointTM} has no false negatives and in Dealer Mode, they achieve online speedups of $2\times$ and $8\times$ over ePPTM, respectively.

Roadmap: The remainder of this paper is organised as follows: In §II we compare our protocols with the state-of-the-art. In §III we introduce trajectory matching and the cryptographic concepts used in *MPCTM*. §IV introduces the design of *MPCTM*, while §V gives a thorough exposition of the experimental results. Finally, in §VI we discuss the relevant deployment aspects and highlights identified edge cases. For the interested reader, we provide a detailed communication complexity analysis of our protocols in Appendix C and security analysis in Appendix D.

II. Related Work

Several works protect location privacy through anonymisation, cloaking, or access control rather than secure matching. Ye et al. [45] construct l-diverse cloaking regions for continuous Location Based Services (LBS) queries, and Svaigen et al. [40] study trajectory anonymity in the drone setting, but neither cryptographically protects the underlying trajectory data. Nkenyereye et al. [32] and Zhang et al. [47] instead assume Public-Key Infrastructure (PKI) and trusted entities. *MPCTM* requires no anonymity set, PKI, or trusted institutional authority: in autonomous mode the parties compute directly, while in dealer mode the dealer may be semi-honestly corrupted.

Closer to our setting, several schemes perform private geometric proximity or collision detection. Xu et al. [44] detect the overlap of two moving circles under MPC, adapting the detection interval to the objects' distance and velocity, but are limited to two-dimensional circular objects whose abstraction introduces imprecision and false positives. Frikken et al. [18] match points and line segments, which is conceptually close to private route comparison, yet remain mainly theoretical and provide no full trajectory-matching implementation. Desai et al. [11] encode trajectories as airspace matrices and detect conflicts by securely summing the entries contributed by each party. This requires a pre-agreed flight space and temporal structure, scales poorly as the matrix grows with the monitored area and resolution, and can miss collisions when trajectories are sampled or aligned differently. We instead model the full trajectory rather than isolated points, line segments, or circular abstractions, representing movement in Cartesian coordinates derived from GPS with explicit timestamps in universal Unix time. This requires no global occupancy matrix, no specific sampling strategy, and no synchronised start, enabling an exact evaluation of the collision predicate without the false negatives induced by coarse spatial or temporal discretisation, and we

provide a proof-of-concept implementation in two different frameworks.

Other UAS-oriented schemes address adjacent problems: PPCA [43] and Tedeschi et al. [41] target privacy-preserving collision avoidance and proximity discovery, Sun et al. [39] perform spatiotemporal matching for device-to-device communication, and Jia et al. [27] study LBS queries using Function Secret Sharing-style techniques. These works are complementary but do not solve full, time-aware trajectory matching over private planned trajectories.

The closest prior works are PPTM [35] and ePPTM [19]. PPTM detects spatiotemporal collisions in a decentralised manner by combining incremental capsule matching with lightweight privacy-preserving proximity testing, requiring neither trusted third parties nor continuous Internet connectivity. ePPTM extends it to sparse trajectory sampling through enhanced responder-side capsule generation and fictitious points, thereby avoiding false negatives under sparse conditions, and adds Full and Truncated modes that trade computational overhead against false positives. Both, however, encode trajectories as capsule-oriented spatial regions whose orientation and refinement patterns reveal structural information such as movement direction, and both derive trajectory identifiers from a fixed spatial encoding tied to the original trajectory geometry, which precludes adaptive or dynamically re-randomised encodings. *MPCTM* replaces capsule-based identifiers by evaluating the trajectory-matching predicate directly over secret-shared inputs. This supports both full and sparse trajectory representations, incorporates time and motion into the secure computation, removes the structural leakage of fixed capsule orientation and recursive spatial refinement, and provides a more flexible basis for higher-dimensional trajectory matching. Table I summarises the resulting differences across the five properties central to privacy-preserving trajectory matching.

III. Preliminaries

In the following, we describe the preliminaries used throughout the paper. We first introduce the notation, followed by the data encoding used for secure processing. We then describe preprocessing-based secret-sharing MPC, and finally present the system and threat models.

A. Notation

We denote vectors and matrices by bold letters, such as $\mathbf{v}$ and $\mathbf{M}$, respectively. For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. Let $\mathbf{1}(\cdot)$ denote the indicator function, which evaluates to 1 if the condition inside holds, and to 0 otherwise. We denote by Π_{Func} an MPC protocol realizing the functionality Func.

B. Data Encoding

On a high level, our protocol for trajectory matching between two distinct parties relies on the secure computation of distances in a multidimensional space based on their respective coordinates. We assume that coordinates are represented in a Cartesian coordinate system obtained via a predetermined map projection. Each party can locally derive these Cartesian coordinates by transforming their sampled geodetic coordinates (e.g., latitude and longitude obtained from GPS measurements) using the agreed-upon projection.

While geodetic coordinates are typically represented as real-valued floating-point numbers, MPC protocols generally operate over finite algebraic structures to enable secure computation [33], [46]. Integer rings enable efficient modular arithmetic due to their natural mapping onto hardware, thereby reducing the computational overhead for the computing parties. In our constructions, we thus rely on MPC protocols defined over the ring $\mathbb{Z}_{2^\ell}$. To accommodate this setting, a real number $\tilde{x} \in \mathbb{R}$ is converted to a fixed-point representation x by setting $x = \lfloor \tilde{x} \cdot 2^k \rfloor \in \mathbb{Z}$, where k denotes the fixed precision parameter and $\lfloor \cdot \rfloor$ denotes the floor function. Given this integer representation, the value can be encoded into the ring using the standard two's complement representation, as formalised in Definition III.1.

Definition III.1 (Data encoding)**.** Let $x \in \mathbb{Z}$, such that $|x| < 2^{\ell-1}$. The encoding z of x in $\mathbb{Z}_{2^\ell}$ is defined as

$$z = \begin{cases} x, & \text{if } x \geq 0 \\ 2^\ell - |x|, & \text{if } x < 0 \end{cases}.$$

We define $\mathsf{Decode}(z) = x$ to be the decoding function transforming a ring element $z \in \mathbb{Z}_{2^\ell}$ into a signed integer $x \in \mathbb{Z}$.

C. Secure Processing

To enable secure processing of integer-encoded trajectory points, we use secret-sharing-based MPC as the main computation backend. The parties represent private coordinates and timestamps in shared form and evaluate the trajectory matching protocol without revealing the underlying values.

1) Secret Sharing: In this work, we rely on a two-party linear secret sharing scheme. A linear secret sharing scheme provides the following operations:

- **Share.** A secret $x \in \mathbb{Z}_{2^\ell}$ is split into two shares, one held by each party P_1 and P_2. Individually, a share does not reveal information about x, while the two shares together allow reconstruction. We use arithmetic sharings over the ring $\mathbb{Z}_{2^\ell}$, denoted by $\langle x \rangle$, and Boolean sharings over a binary field extension $\mathbb{Z}_2^\ell$, denoted by $\langle x \rangle^B$.
- **Lin.** The sharing scheme is linear. Given shared values $\langle x \rangle$ and $\langle y \rangle$, and public constants $\alpha, \beta, \gamma \in \mathbb{Z}_{2^\ell}$, the parties can locally compute a sharing of $\alpha x + \beta y + \gamma$. Thus, additions, subtractions, and multiplications by public constants do not require interaction.
- **Mult.** Given two shared values $\langle x \rangle$ and $\langle y \rangle$, the parties compute a sharing of $x \cdot y$ using Beaver multiplication triples [3]. A triple consists of shared random values $\langle a \rangle$, $\langle b \rangle$, and $\langle c \rangle$, where $a \cdot b = c$, generated during a setup phase. In this work, we rely on both Oblivious Transfer (OT)- and Homomorphic Encryption (HE)-based generation of multiplication triples, described below. To

TABLE I: Qualitative comparison of privacy-preserving trajectory matching schemes. *Full trajectory matching*: complete planned trajectories are compared, rather than individual locations or simplified geometric objects. *Spatiotemporal matching*: both spatial proximity and temporal overlap are considered. *No leakage beyond output*: nothing is revealed other than the prescribed functionality output, in particular, the encoding itself leaks no direction-dependent structure. *Dynamic trajectory encoding*: trajectories are encoded in a time- and motion-aware manner rather than through a fixed static representation. *Sampling-agnostic*: correctness does not rely on synchronised timestamps, identical sampling rates, a common start time, or sufficiently dense trajectory samples. ● indicates the property is fulfilled and ○ that it is not fulfilled.

Contribution	Cryptography Technique	Full Trajectory Matching	Spatiotemporal Matching	No leakage beyond output	Dynamic encoding of Trajectories	Sampling-agnostic
Xu et al. [44]	MPC	○	○	○	●	●
Frikken et al. [18]	MPC	○	○	●	○	●
Desai et al. [11]	MPC	●	○	○	○	○
PPCA [43]	El-Gamal	○	●	○	○	○
Sun et al. [39]	PSI+BF	●	○	○	○	○
Tedeschi et al. [41]	PSI	○	○	○	●	○
PPTM [35]	PSI	●	●	○	○	●
ePPTM [19]	PSI	●	●	○	○	●
MPCTM	**MPC**	●	●	●	●	●

preserve fixed-point encoding, non-interactive truncation is applied after every multiplication [30].

- **Recon.** To reconstruct a value, the parties exchange their shares and combine them according to the reconstruction rule of the sharing scheme.

Additionally, we will require a comparison functionality, that upon input of an encoding $z \in \mathbb{Z}_{2^\ell}$ outputs whether $\mathsf{Decode}(z) < 0$. We instatiate this using a less-than-zero functionality $\mathsf{LTZ}(x)$ defined as follows:

$$\mathsf{LTZ}(z) = \begin{cases} 0, & z \in \left[0, 2^{\ell-1}\right) \\ 1, & z \in \left[2^{\ell-1}, 2^\ell - 1\right] \end{cases}.$$

In this work we employ approaches based on mixed circuit computation for evaluation of non-linear layers in the style of [10], [33].

a) OT-based triple generation: Oblivious transfer is a two-party primitive where a sender holds two values m_0, m_1, and a receiver with choice bit b learns only m_b. The sender does not learn b, and the receiver does not learn m_{1-b} [26]. This can be used to multiply shared values by reducing the cross-term computation to OT instances. For example, when x and y are additively shared as $x = \langle x \rangle_1 + \langle x \rangle_2$ and $y = \langle y \rangle_1 + \langle y \rangle_2$, each party can locally compute the terms $\langle x \rangle_i \langle y \rangle_i$, while the mixed terms $\langle x \rangle_1 \langle y \rangle_2$ and $\langle x \rangle_2 \langle y \rangle_1$ are computed securely using OT-based correlated randomness. This produces additive shares of xy without revealing the secret shares.

b) HE-based triple generation: Homomorphic encryption [15] allows one party to compute on values encrypted by the other party. Let $\mathsf{Enc}_{\mathsf{pk}}(\cdot)$ denote encryption under a public key pk, and let $\mathsf{Dec}_{\mathsf{sk}}(\cdot)$ denote decryption under the corresponding secret key sk. Given ciphertexts $\mathsf{Enc}_{\mathsf{pk}}(x)$ and $\mathsf{Enc}_{\mathsf{pk}}(y)$, the homomorphic evaluation algorithm supports arithmetic operations such that decrypting the result gives the corresponding plaintext operation, e.g., $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(x) + \mathsf{Enc}_{\mathsf{pk}}(y)) = x + y$ and $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(x) \cdot y) = xy$, depending on the supported operations of the scheme.

HE can be used to securely multiply values held by two parties. Suppose P_1 holds x and P_2 holds y, and the goal is to obtain additive shares of xy. Party P_1 generates an HE key pair $(\mathsf{pk}, \mathsf{sk})$, sends pk and $\mathsf{Enc}_{\mathsf{pk}}(x)$ to P_2, and keeps sk private. Party P_2 samples a random mask r, homomorphically computes $\mathsf{Enc}_{\mathsf{pk}}(xy - r) = \mathsf{Enc}_{\mathsf{pk}}(x) \cdot y - \mathsf{Enc}_{\mathsf{pk}}(r)$, and sends the resulting ciphertext back to P_1. After decryption, P_1 obtains $xy - r$, while P_2 keeps r. Thus, the parties hold additive shares of xy, since $(xy - r) + r = xy$, without revealing x or y. This mechanism can be used during preprocessing to generate Beaver triples $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$, where $c = a \cdot b$. In our implementation, we instantiate HE-based triple generation using the RLWE-based protocol of Cheetah [23].

c) Dealer-based triple generation: When a semi-honest, independent third party is available, it can act as a dealer by generating input-independent preprocessing material for the computing parties. Specifically, the dealer locally generates all required correlated randomness, secret-shares it, and distributes the resulting shares among the computing parties.

D. System Model

In our system model, we consider two UASs operated by mutually distrustful service providers and manufactured by different vendors. Each UAS is deployed with a pre-planned trajectory consisting of a sequence of trajectory points and their associated timestamps. Depending on the deployment environment, the trajectory points may be defined in three dimensions, as in the case of unmanned submarines or unmanned aerial vehicles, or in two dimensions, as in the case of ground vehicles operating in environments without traffic rules or autonomous surface vessels.

We assume that each UAS is equipped with radio communication modules that allow it to periodically emit beacons and identify nearby UASs within radio visibility, depicted in Fig. 1 by the red and blue circles. After discovering one another, the two parties establish a direct connection using WiFi Direct. They subsequently establish a secure communication

channel using TLS. Although TLS provides authentication, integrity, and confidentiality for the communication channel, it does not prevent privacy leakage caused by information intentionally disclosed during the execution of the collision-detection protocol.

At the same time, WiFi-based communication modules deployed on UASs may support communication over distances of up to 25 km [42], [13]. Therefore, being within radio visibility does not necessarily imply an immediate collision risk. Nevertheless, once two UASs discover each other, they need to determine whether their pre-programmed trajectories contain conflicting segments. This determination must be performed without disclosing trajectory points that are unrelated to the potential collision.

Within this general setting, we consider two deployment models for privately resolving trajectory conflicts. The two models differ in how the input-independent MPC preprocessing material is generated and provisioned to the parties.

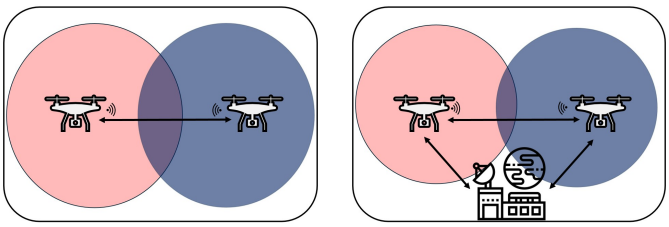
a) Autonomous model: Fig. 1(a) depicts the autonomous model, in which no supporting infrastructure is available when the two UASs enter radio visibility. As described above, the parties first discover each other and establish a secure connection. Afterwards, they jointly execute both the preprocessing phase and the online phase of the MPC protocol to identify potential collisions between their trajectories.

In this model, the main communication overhead is introduced by the generation of Beaver triples between the two parties. The required number of triples grows with $N \times N'$, where N denotes the number of trajectory points belonging to one party and N' denotes the number of trajectory points belonging to the other party.

b) Dealer-assisted model: Fig. 1(b) presents the dealer-assisted model. In this setting, we assume that both parties are within communication range of a supporting infrastructure component, such as a ground station or roadside unit [6], [9]. This infrastructure acts as a dealer and generates the cryptographic material required for the preprocessing phase. Consequently, the two UASs only need to establish a secure connection and execute the online phase of the protocol to identify collisions between their trajectories. We assume that the dealer is semi-honest. In particular, the dealer follows the prescribed protocol but may attempt to infer information from the data available to it. However, the dealer does not receive the private trajectory inputs of either party. Its role is limited to generating and distributing the correlated values required for the Beaver triples. The only information revealed to the dealer is the number of trajectory points held by each party.

E. Threat Model

We assume a semi-honest adversary, denoted by $\mathcal{A}$, whose goal is to learn private trajectory information of another party involved in the collision-detection protocol. We adopt the semi-honest model because malicious security at the MPC layer alone cannot ensure that the trajectory submitted by a UAS corresponds to its pre-planned or physically followed trajectory. A maliciously secure MPC protocol can ensure



(a) Autonomous model in UAS-to-UAS setting.

(b) Dealer-based model in UASs with semi-honest dealer/Ground station setting.

Fig. 1: Visualisation of our two system models.

that the agreed computation is correctly evaluated on the inputs supplied by the parties and can prevent deviations from the prescribed protocol steps. However, it cannot prevent a compromised UAS from supplying arbitrary trajectory points, submitting a trajectory different from the one it intends to follow, or physically deviating from the submitted trajectory after the protocol execution. Addressing such behavior requires input-verification and physical-compliance mechanisms outside the MPC protocol, such as trusted sensing, remote attestation, or authenticated flight plans. These mechanisms are outside the scope of this work and are discussed further in Sec. VI.

Similarly, physical surveillance and physical-layer localisation attacks are orthogonal to our work and are outside its scope. In particular, our work does not aim to protect against adversaries that can independently determine the position or trajectory of a UAS through visual observation, radar, radio-frequency localisation, signal-strength measurements, time-of-flight measurements, or Doppler-effect measurements. Such attacks can be performed independently of the collision-detection protocol and cannot be prevented by MPC. Instead, our objective is to ensure that participation in the protocol does not reveal additional private trajectory information beyond the intended protocol output.

The main objective of $\mathcal{A}$ is to retrieve trajectory-related information of a target UAS. Such information includes the locations along the path, the times at which specific locations are reached, the start and end times of the mission, and the overall mission duration. Access to such information can enable severe attacks, such as tracking the UAS, launching targeted jamming attacks, capturing the UAS or the goods it delivers, disrupting its operations, and causing significant financial losses. Therefore, our goal is to protect the involved UASs' trajectory and time-related information.

In the autonomous system model, we assume that the adversary may corrupt one of the participating UASs. The corrupted party follows the prescribed protocol without deviation, but may use its local input, randomness, internal protocol state, received messages, and output to infer information about the other UAS's trajectory.

In the dealer-assisted system model, we consider two separate corruption cases: (i) the adversary corrupts one of the

participating UASs, while the dealer and the other UAS remain honest or (ii) the adversary corrupts the dealer, while both participating UASs remain honest. We do not consider collusion between the dealer and either participating UAS. Therefore, the adversary does not simultaneously obtain the internal state of the dealer and that of a participating UAS. This non-collusion assumption is a trust requirement of the dealer-assisted system model.

We assume that the communication channel between the parties is protected using WiFi Direct and TLS, as described in the system model. Therefore, external eavesdroppers may observe the presence of Radio Frequency (RF) activity and infer that two devices are within radio visibility, but they cannot access the contents of the protocol messages. However, TLS only protects the communication channel and does not prevent privacy leakage caused by the information intentionally exchanged during the protocol. Our protocol therefore aims to ensure that the messages exchanged during collision detection reveal only the intended output, namely whether a collision exists and, depending on the protocol variant, the corresponding colliding points.

IV. Secure Trajectory Matching

In this section, we present the high-level idea behind our trajectory-matching protocol. We consider two UASs, each representing its trajectory as a discrete set of sampled points representing the spatial position at any given time. In contrast to previous work, our protocol does neither impose a particular sampling strategy, nor relies on synchronised sampling rate. This limits the leakage of SOTA protocols in this domain as described in Note 1.

Note 1. *State-of-the-art approaches, such as [41], [39], assume that the parties agree in advance on a common sampling strategy. This may involve sampling at fixed temporal intervals, such as every x seconds, or at fixed spatial intervals, such as every x meters. Although the underlying protocol protects the sampled data, the length of the input vectors may reveal information about the total distance travelled by the other party. A straightforward mitigation is to pad the input vectors with dummy values, but this introduces additional computational overhead. In contrast, our solution uses a sampling-agnostic protocol that allows each party to choose its sampling strategy and rate independently.*

A. Modelling of Trajectory Representation

The goal of this work is to allow the UASs to securely determine whether their trajectories collide, without revealing any additional information about their paths. More precisely, each party should learn only whether a potential collision occurs, and, if so, the points at which such a collision is detected. No other information about the other party's trajectory should be revealed.

Given the discrete point sets describing the approximate trajectories of the two UASs, the basic idea is to compare points from the two trajectories pairwise. If any point from one trajectory lies within a predefined distance bound of any point from the other trajectory, we identify this as a potential collision.

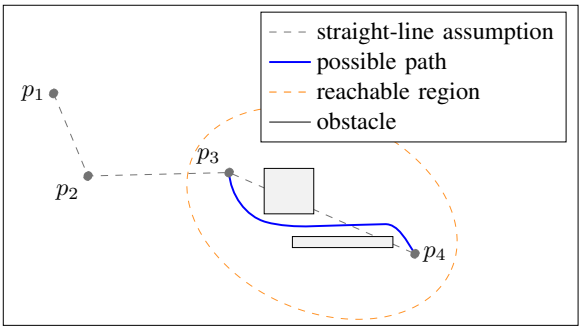


Fig. 2: Illustration of a possible trajectory given a discrete set of sampled positions. Obstacles may block the assumed straight-line connection, requiring a curved path within the reachable elliptic region.

This approach raises an important modelling question: how should the position of a UAS be described between two consecutive sampled points? For example, assume that a UAS samples its trajectory every five seconds. At time t, the UAS is known to be at a given position, and at time $t + 5$ it is required to reach the next sampled position. However, the exact path followed during this time interval is not captured by the sampled trajectory, even though it may be known to the UAS. Therefore, without additional assumptions, we cannot conclude that the UAS follows a straight line, or any other fixed path, between the two sampled points. This ambiguity is illustrated in Figure 2.

To address this uncertainty, we model the UAS's possible behaviour between two consecutive sampled points. Instead of considering only the sampled positions themselves, we consider the spatial region that the UAS could potentially reach between them, under the constraint that it must arrive at the next sampled position at the specified time. We construct this reachable region using the UAS's maximum velocity, which captures the farthest distance the UAS could travel during the given time interval. The constructed reachable region between any two points is an ellipse with foci at the two points. This allows the protocol to account for all possible intermediate positions that the UAS may occupy between sampled points, rather than relying solely on the sampled trajectory points themselves. We formally define such elliptic region in Definition A.1.

To evaluate whether two paths collide when their reachable regions are represented by ellipses, we would need to determine whether any ellipse from one path overlaps with any ellipse from the other path. However, checking the overlap between two ellipses is a non-trivial geometric problem. In general, it requires solving an optimisation problem, which is expensive and therefore undesirable in a secure computation setting.

To make the collision test more efficient, we approximate each ellipse by an Oriented Bounding Rectangle (OBR). This

rectangle is chosen as the smallest rectangle fully containing the ellipse, and therefore also containing all positions that the UAS may reach within the corresponding interval. We formally define the OBR that encloses the reachable elliptic region in Definition A.2. Collision detection can then be performed by checking for intersections between these rectangles instead of directly comparing ellipses.

This simplification comes at the cost of precision. Since the bounding rectangle contains the ellipse but does not match it exactly, it also includes additional space that is not actually reachable by the UAS. This extra region may lead to false positives: the protocol may report a potential collision between two rectangles even when the underlying ellipses do not overlap. Nevertheless, the approximation is conservative, since it does not exclude any reachable position of the UAS.

Given the spatial representation of the OBRs, we can determine whether two such rectangles overlap using the Separating Axis Theorem (SAT) [21]. The SAT provides a geometric criterion for detecting intersections between convex shapes: two rectangles do not overlap if and only if there exists an axis along which their projections are disjoint.

We use this principle to derive an overlap test for pairs of oriented bounding rectangles. In particular, we define the overlap condition between any two rectangles in terms of their projections onto the relevant separating axes. This SAT-based overlap criterion is formalised in Lemma IV.1.

Lemma IV.1 (Overlap of oriented bounding rectangles)**.** *Let*

$$\rho_1 = (\mathbf{m}_1, \mathbf{u}_1, \mathbf{v}_1, a_1, b_1), \qquad \rho_2 = (\mathbf{m}_2, \mathbf{u}_2, \mathbf{v}_2, a_2, b_2)$$

be two oriented rectangles in $\mathbb{R}^2$, where $\mathbf{m}_i$ is the center, $\mathbf{u}_i, \mathbf{v}_i$ are orthonormal direction vectors, and $a_i, b_i > 0$ are the corresponding half-side lengths. We define the distance between the two center points to be $\mathbf{d} = \mathbf{m}_2 - \mathbf{m}_1$.

For an axis $\mathbf{n} \in \mathbb{R}^2$, define the projection radius of rectangle ρ_i onto $\mathbf{n}$ as

$$r_i(\mathbf{n}) = a_i|\mathbf{u}_i \cdot \mathbf{n}| + b_i|\mathbf{v}_i \cdot \mathbf{n}|. \tag{1}$$

Then ρ_1 and ρ_2 overlap if and only if, for every $\mathbf{n} \in \{\mathbf{u}_1, \mathbf{v}_1, \mathbf{u}_2, \mathbf{v}_2\}$, we have

$$|\mathbf{d} \cdot \mathbf{n}| - r_1(\mathbf{n}) - r_2(\mathbf{n}) \leq 0\,. \tag{2}$$

Equivalently, the rectangles overlap if and only if the following four inequalities hold:

$$\begin{aligned}
|\mathbf{d} \cdot \mathbf{u}_1| &\leq a_1 + a_2|\mathbf{u}_2 \cdot \mathbf{u}_1| + b_2|\mathbf{v}_2 \cdot \mathbf{u}_1|,\\
|\mathbf{d} \cdot \mathbf{v}_1| &\leq b_1 + a_2|\mathbf{u}_2 \cdot \mathbf{v}_1| + b_2|\mathbf{v}_2 \cdot \mathbf{v}_1|,\\
|\mathbf{d} \cdot \mathbf{u}_2| &\leq a_2 + a_1|\mathbf{u}_1 \cdot \mathbf{u}_2| + b_1|\mathbf{v}_1 \cdot \mathbf{u}_2|,\\
|\mathbf{d} \cdot \mathbf{v}_2| &\leq b_2 + a_1|\mathbf{u}_1 \cdot \mathbf{v}_2| + b_1|\mathbf{v}_1 \cdot \mathbf{v}_2|.
\end{aligned} \tag{3}$$

If all four inequalities hold, the rectangles overlap or touch. If at least one inequality fails, the rectangles are separated.

Originally, each trajectory was represented as a discrete set of sampled points, where every point was described by Cartesian spatial coordinates and a timestamp. To account for all possible UAS positions between consecutive sampled points, we replace this point-based representation with a rectangle-based one. In this representation, a trajectory is described by a sequence of rectangles, where each rectangle corresponds to the reachable region between two consecutive points on the original path.

Each oriented bounding rectangle is specified by its geometric centre, two unit vectors defining its orientation, and two half-side lengths defining its size. Thus, instead of representing a trajectory only by sampled positions, we represent it as a set of OBRs associated with the corresponding time intervals. We formally define this rectangle-based trajectory representation in Definition IV.2.

Definition IV.2 (OBR-based trajectory)**.** Let $\mathcal{T}^{(j)} = \big((\mathbf{r}_i^{(j)}, t_i^{(j)})\big)_{i=1}^{N}$ be the sampled trajectory of party j. For each consecutive pair $\big((\mathbf{r}_i^{(j)}, t_i^{(j)}), (\mathbf{r}_{i+1}^{(j)}, t_{i+1}^{(j)})\big)$, let $\mathcal{E}_i^{(j)}$ be the corresponding reachable ellipse and let $\rho_i^{(j)} = \Big(\mathbf{m}_i^{(j)}, \mathbf{u}_i^{(j)}, \mathbf{v}_i^{(j)}, a_i^{(j)}, b_i^{(j)}\Big)$ be its oriented bounding rectangle representation. We define the rectangle-based trajectory of party j as

$$\mathcal{R}^{(j)} = \big(\rho_i^{(j)}, t_i^{(j)}\big)_{i=1}^{N}.$$

B. OBR-Based Trajectory Matching

Protocol 1 describes the modelled rectangle-based trajectory-matching procedure. Each party first converts its trajectory into the oriented bounding rectangle representation and secret-shares this representation with the other party (Step 1). The parties then evaluate rectangle overlap for every pair of rectangles, one from each trajectory, using the SAT-based procedure from Lemma IV.1.

For each pair of bounding rectangles, the parties begin by computing the coordinate-wise difference between their centres, $\mathbf{d} = \mathbf{m}_2 - \mathbf{m}_1$ (Step 6), following the notation of Lemma IV.1. Then, for every directional unit vector $\mathbf{n} \in \{\mathbf{u}_1, \mathbf{v}_1, \mathbf{u}_2, \mathbf{v}_2\}$ they compute the dot product $\mathbf{d} \cdot \mathbf{n}$ and its absolute value (Step 8). For each of the two rectangles, the parties also compute the corresponding projection radius according to Equation 1 (Steps 9–10). Using these values, they evaluate whether a separating axis exists in direction $\mathbf{n}$ (Step 11), as specified in Equation 2, and perform the corresponding comparison (Step 12).

For each pair of rectangles, apart from spatial collision, the parties verify the collision in time. We define $\Delta t = \max\big(|t_i^{(1)} - t_{i+1}^{(1)}|, |t_j^{(2)} - t_{j+1}^{(2)}|\big)$ as the maximum of the two time intervals between the sampled points used to construct the corresponding rectangles. The parties then determine whether the rectangles collide in time by checking whether the difference between their timestamps lies within this bound (Steps 14–17).

The output of the comparison protocol is a single secret-shared bit. To reduce communication, the remaining computation is therefore performed directly over secret-shared bits, rather than converting these bits to an arithmetic domain. By Lemma IV.1, two rectangles overlap if and only if there is no separating axis among the four candidate axes. Equivalently, a collision between two rectangles occurs if $(\bigwedge_{\mathbf{n}} s_{\mathbf{n}}) \wedge s_t = 1$, where $s_{\mathbf{n}} = \mathbf{1}(|\mathbf{d} \cdot \mathbf{n}| - r_1(\mathbf{n}) - r_2(\mathbf{n}) \leq 0)$ (Step 18). Here, $s_{\mathbf{n}}$

indicates whether the projections of the two rectangles overlap along axis $\mathbf{n}$.

After evaluating this condition for all pairs of rectangles, the parties obtain secret-shared collision indicators for every rectangle pair. Finally, these bits are reconstructed, allowing the parties to learn whether a collision occurs and, if so, which rectangle pairs are involved, without revealing any additional information about their trajectories.

Protocol 1 Π_{OBRTM}: OBR-based Trajectory Matching

Input: $\mathcal{R}^{(1)} = \left(\rho_i^{(1)}, t_i^{(1)}\right)_{i=1}^{N}$, $\mathcal{R}^{(2)} = \left(\rho_j^{(2)}, t_j^{(2)}\right)_{j=1}^{N'}$, $t_{\mathsf{max}}^{(1)}$, $t_{\mathsf{max}}^{(2)}$.

Output: $\{s_{ij}\}_{(i,j)\in[N]\times[N']}$.

$\langle \mathcal{R}^{(1)} \rangle \leftarrow \Pi_{\mathsf{Share}}(\mathcal{R}^{(1)})$, $\langle \mathcal{R}^{(2)} \rangle \leftarrow \Pi_{\mathsf{Share}}(\mathcal{R}^{(2)})$
$\langle t_{\mathsf{max}}^{(1)} \rangle \leftarrow \Pi_{\mathsf{Share}}(t_{\mathsf{max}}^{(1)})$, $\langle t_{\mathsf{max}}^{(2)} \rangle \leftarrow \Pi_{\mathsf{Share}}(t_{\mathsf{max}}^{(2)})$
$\langle \Delta t \rangle = \Pi_{\mathsf{Max}}\left(\langle t_{\mathsf{max}}^{(1)} \rangle, \langle t_{\mathsf{max}}^{(2)} \rangle\right)$
$\langle (\Delta t)^2 \rangle = \langle \Delta t \rangle \cdot \langle \Delta t \rangle$
for $(i,j) \in [N] \times [N']$ **do**
  $\langle \mathbf{d}_{ij} \rangle = \langle \mathbf{m}_j^{(2)} \rangle - \langle \mathbf{m}_i^{(1)} \rangle$
  for $\mathbf{n} \in \{\mathbf{u}_i^{(1)}, \mathbf{v}_i^{(1)}, \mathbf{u}_j^{(2)}, \mathbf{v}_j^{(2)}\}$ **do**
    $\langle \ell_{\mathbf{n}_{ij}} \rangle = |\langle \mathbf{d}_{ij} \rangle \cdot \langle \mathbf{n} \rangle|$
    $\langle r_i^{(1)}(\mathbf{n}) \rangle = \langle a_i^{(1)} \rangle \left|\langle \mathbf{u}_i^{(1)} \rangle \langle \mathbf{n} \rangle\right| + \langle b_i^{(1)} \rangle \left|\langle \mathbf{v}_i^{(1)} \rangle \langle \mathbf{n} \rangle\right|$
    $\langle r_j^{(2)}(\mathbf{n}) \rangle = \langle a_j^{(2)} \rangle \left|\langle \mathbf{u}_j^{(2)} \rangle \langle \mathbf{n} \rangle\right| + \langle b_j^{(2)} \rangle \left|\langle \mathbf{v}_j^{(2)} \rangle \langle \mathbf{n} \rangle\right|$
    $\langle \Delta_{\mathbf{n}_{ij}} \rangle = \langle \ell_{\mathbf{n}_{ij}} \rangle - \langle r_i^{(1)}(\mathbf{n}) \rangle - \langle r_j^{(2)}(\mathbf{n}) \rangle$
    $\langle s_{\mathbf{n}_{ij}} \rangle^B \leftarrow \Pi_{\mathsf{LTZ}}(\langle \Delta_{\mathbf{n}_{ij}} \rangle)$
  end for
  $\langle t_{ij} \rangle = \langle t_i^{(1)} \rangle - \langle t_j^{(2)} \rangle$
  $\langle (t_{ij})^2 \rangle = \langle t_{ij} \rangle \cdot \langle t_{ij} \rangle$
  $\langle \Delta_{t_{ij}} \rangle = \langle (t_{ij})^2 \rangle - \langle (\Delta t)^2 \rangle$
  $\langle s_{t_{ij}} \rangle^B \leftarrow \Pi_{\mathsf{LTZ}}(\langle \Delta_{t_{ij}} \rangle)$
  $\langle s_{ij} \rangle^B = \bigwedge_{\mathbf{n}\in\{\mathbf{u}_i^{(1)}, \mathbf{v}_i^{(1)}, \mathbf{u}_j^{(2)}, \mathbf{v}_j^{(2)}\}} \langle s_{\mathbf{n}_{ij}} \rangle^B \wedge \langle s_{t_{ij}} \rangle^B$
  $s_{ij} \leftarrow \Pi_{\mathsf{Recon}}(\langle s_{ij} \rangle^B)$
end for

C. Point-based Trajectory Matching

Previously we presented a trajectory matching protocol based on a dynamic, capsule-like modelling of trajectories. While this approach accurately captures the behaviour of a UAS between discretely sampled points, it incurs a significant computational cost. In particular, it requires numerous multiplications and comparisons, which translate into communication overhead in the MPC setting. Moreover, the protocol does not scale well to higher-dimensional spaces. The number of required operations grows exponentially with the number of dimensions. For instance, in three-dimensional space, the protocol requires 15 inequality checks. This significantly affects the overall performance of the protocol.

The protocol represents each trajectory as a sequence of sampled points. Each point contains spatial Cartesian coordinates and a corresponding timestamp. We formalise this representation in Definition IV.3.

Definition IV.3 (Trajectory)**.** Let $P_i^{(j)}$ be the i-th sampled point of party j at time $t_i^{(j)}$, with spatial Cartesian coordinates $\mathbf{r}_i^{(j)} = \left(x_i^{(j)}, y_i^{(j)}, z_i^{(j)}\right) \in \mathbb{R}^3$. We define the discrete trajectory of party j with N sampled points as

$$\mathcal{T}^{(j)} = \left((\mathbf{r}_i^{(j)}, t_i^{(j)})\right)_{i=1}^{N}.$$

Given two trajectories $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$, the protocol compares every sampled point of the first trajectory with every sampled point of the second trajectory. For a pair of points $P_i^{(1)}$ and $P_j^{(2)}$, the parties check whether the points are sufficiently close in each spatial coordinate and in time. Let Δd_x, Δd_y and Δd_z denote the spatial thresholds for the three Cartesian coordinates, and let Δd_t denote the time threshold. The coordinate-wise comparison is then defined as

$$s_{n_{ij}} = \mathbf{1}(|n_i^{(1)} - n_j^{(2)}| < \Delta d_n)\,, \forall n \in \{x, y, z, t\}\,. \tag{4}$$

A collision is reported for the pair (i, j) only if all four comparisons are satisfied. Equivalently, the protocol reports a collision whenever UAS 2 is located within the rectangular collision region around UAS 1 during the time interval $t_i^{(1)} \pm \Delta d_t$. This region is axis-aligned and determined by the thresholds Δd_x, Δd_y and Δd_z. We illustrate the resulting 3D trajectory matching approach in Appendix B.

The threshold bounds are determined dynamically and do not depend on a particular sampling strategy of either UAS. Instead, they are derived from the maximum velocity of the UASs and half of the largest time interval between two consecutive sampled points. Intuitively, this gives an upper bound on the maximum distance that the UASs can travel from a sampled point within the considered time window. The procedure is described in Protocol 2. Each party secret shares its derived local distance bounds. The parties then compute the total distance as a sum, adding a public parameter ϵ for safety margin.

Protocol 2 Π_{Bounds}: Distance bounds δ

Input: $d^{(1)}$, $t^{(1)}$, $d^{(2)}$, $t^{(2)}$; public margin ϵ.

Output: $\langle \delta^2 \rangle = \{\Delta^2 d_x, \Delta^2 d_y, \Delta^2 d_z, \Delta^2 d_t\}$.

$\langle d^{(1)} \rangle \leftarrow \Pi_{\mathsf{Share}}(d^{(1)})$, $\langle d^{(2)} \rangle \leftarrow \Pi_{\mathsf{Share}}(d^{(2)})$
$\langle t^{(1)} \rangle \leftarrow \Pi_{\mathsf{Share}}(t^{(1)})$, $\langle t^{(2)} \rangle \leftarrow \Pi_{\mathsf{Share}}(t^{(2)})$
$\langle d_{\mathsf{total}} \rangle = \langle d^{(1)} \rangle + \langle d^{(2)} \rangle + \epsilon$
$\langle t_{\mathsf{total}} \rangle = \langle t^{(1)} \rangle + \langle t^{(2)} \rangle$
$\langle \delta^2 \rangle = \{\langle d_{\mathsf{total}}^2 \rangle, \langle d_{\mathsf{total}}^2 \rangle, \langle d_{\mathsf{total}}^2 \rangle, \langle t_{\mathsf{total}}^2 \rangle\}$

The MPC evaluation of the simplified trajectory matching protocol is described in Protocol 3. First, the parties secret share their trajectories (Step 1). They then obtain the coordinate thresholds dynamically, as described previously (Step 2). For each pair of sampled points, the parties compute the coordinate differences and compare them with the corresponding thresholds according to Equation 4.

To avoid computing absolute values inside the MPC protocol, we use the standard square-based comparison. For integers e and $f > 0$, the condition $|e| < f$ is equivalent to $e^2 < f^2$. Thus, each coordinate comparison can be evaluated

by squaring the coordinate difference and comparing it with the squared threshold following Equation 4 (Steps 5–8).

Finally, the parties compute $\bigwedge_{n\in\{x,y,z,t\}} s_n = 1$, where s_n denotes the bit determining a collision in a coordinate n (Step 10). The parties then reveal the resulting vector of collision bits, which indicates for which pairs of sampled points a collision was detected (Step 11). The overall trajectories are considered to collide if at least one of these bits is equal to one.

Protocol 3 Π_{PointTM}: Point-based Trajectory Matching

Input: $\mathcal{T}^{(1)} = \left((\mathbf{r}_i^{(1)}, t_i^{(1)})\right)_{i=1}^{N}$, $\mathcal{T}^{(2)} = \left((\mathbf{r}_j^{(2)}, t_j^{(2)})\right)_{j=1}^{N'}$.
Output: $\{s_{ij}\}_{(i,j)\in[N]\times[N']}$.

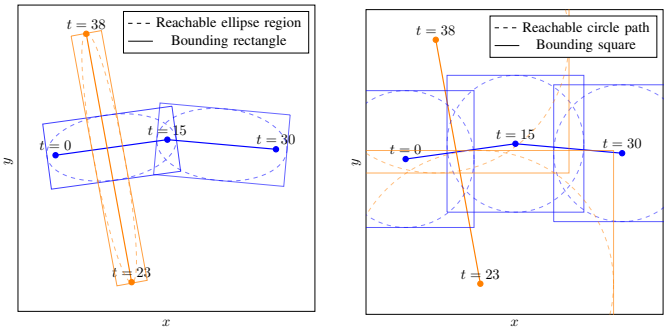
1: $\langle \mathcal{T}^{(1)} \rangle \leftarrow \Pi_{\mathsf{Share}}(\mathcal{T}^{(1)})$, $\langle \mathcal{T}^{(2)} \rangle \leftarrow \Pi_{\mathsf{Share}}(\mathcal{T}^{(2)})$
2: $\langle \delta^2 \rangle \leftarrow \Pi_{\mathsf{Bounds}}()$
3: **for** $(i, j) \in [N] \times [N']$ **do**
4: **for** $n \in \{x, y, z, t\}$ **do**
5: $\langle n_{ij} \rangle = \langle n_i^{(1)} \rangle - \langle n_j^{(2)} \rangle$
6: $\langle {n_{ij}}^2 \rangle = \langle n_{ij} \rangle \cdot \langle n_{ij} \rangle$
7: $\langle \Delta_{c_{ij}} \rangle = \langle {n_{ij}}^2 \rangle - \langle \delta_n^2 \rangle$
8: $\langle s_{n_{ij}} \rangle^B \leftarrow \Pi_{\mathsf{LTZ}}(\langle \Delta_{c_{ij}} \rangle)$
9: **end for**
10: $\langle s_{ij} \rangle^B = \bigwedge_{n\in\{x,y,z,t\}} \langle s_{n_{ij}} \rangle^B$
11: $s_{ij} \leftarrow \Pi_{\mathsf{Recon}}(\langle s_{ij} \rangle^B)$
12: **end for**

This protocol substantially reduces the number of operations required in MPC. The price for this efficiency is a loss of geometric precision. In the Modelled Trajectory Matching protocol, the region occupied by a UAS between two consecutive sampled points is constrained by both endpoints of the segment. That is, the UAS is assumed to be at P_i at time t_i, and at P_{i+1} at time t_{i+1}. The resulting reachable region is therefore elliptic, since every feasible intermediate position must be reachable from P_i and must still allow the UAS to reach P_{i+1} on time.

In contrast, the simplified protocol drops the constraint imposed by the next sampled point. Each sampled point is considered independently, together with a time tolerance and a maximum speed bound. Thus, the protocol asks only how far the UAS could have moved away from its current sampled point within the allowed time window. This gives a larger reachable region than in the endpoint-constrained model. The difference between the two models is illustrated in Figure 3.

Moreover, the protocol evaluates this condition coordinate-wise. As a result, it tests membership in an axis-aligned rectangular region rather than in the exact reachable region. Consequently, some pairs of points may be marked as collisions even though such a collision would be infeasible if the UASs followed their announced paths honestly. These cases correspond to false positives introduced by the simplification. We elaborate on the accuracy of our protocols in Section V-B.

a) **Communication Complexity:** Both protocols are dominated by the $N \cdot N'$ pairwise comparisons, which are mutually independent and evaluated as a single SIMD batch.



(a) Oriented bounding rectangle-based model.

(b) Simplified coordinate-wise model.

Fig. 3: Comparison of the reachable regions used by the modelled endpoint-constrained protocol and the simplified point-based protocol.

The online communication is therefore linear in $N \cdot N'$, while the round complexity is constant in N and N' and depends only on the depth r_{cmp} of the comparison circuit. Per pair of points, Π_{OBRTM} evaluates 8 inner products, 9 multiplications, 8 bit injections, 13 comparisons and 4 AND gates in $7+2\,r_{\mathsf{cmp}}$ rounds, whereas Π_{PointTM} reduces this, for trajectories in three spatial dimensions, to four squarings, four comparisons and three AND gates in $4 + r_{\mathsf{cmp}}$ rounds. Our analysis predicts that Π_{PointTM} lowers the online communication of Π_{OBRTM} by a factor of $4.56\times$ in the function-dependent and $4.70\times$ in the function-independent instantiation, matching the $4.92\times$ and $4.93\times$ observed in Table II and thereby validating the cost model. A full derivation, including the per-primitive costs of both backends, is given in Appendix C.

b) **Security:** The security of our protocols follows from that of their underlying cryptographic building blocks. Both Π_{OBRTM} and Π_{PointTM} apply a sequence of universally composable [7] MPC operations to secret-shared inputs and reconstruct a single value, namely the collision bits $\{s_{ij}\}$. Every intermediate value therefore remains secret shared, and the guarantees of the building blocks carry over to the complete protocol through UC composition. Let $\mathcal{F}_{\mathsf{OBRTM}}$ and $\mathcal{F}_{\mathsf{PointTM}}$ denote the ideal functionalities that, on input the two parties' trajectories, return exactly the collision bits prescribed by Lemma IV.1 and Equation 4. The formal functionality definitions, the simulator, and the proof are given in Appendix D.

V. Evaluation

In the following we describe our evaluation setting, together with experimental results of our protocols compared to the state-of-the-art ePPTM.

A. Proof-of-concept Implementation

We evaluate the performance of our protocols using two different MPC backends, demonstrating their adaptability and versatility. The two implementations differ in the underlying framework and in the secure computation protocols they use.

First, we implement our protocols in MP-SPDZ [28], a framework for benchmarking variety of MPC protocols. This

implementation demonstrates the performance of our protocols using standard and well-established MPC techniques with function-independent preprocessing. The two-party protocol used is based on SPDZ2k [8], stripped of all checks for malicious behavior. This results in an additive secret-sharing-based protocol with an OT-based, function-independent preprocessing phase.

Second, we implement our protocols in HPMPC [22], a framework designed for high-throughput evaluation of machine learning algorithms. This implementation uses an optimised function-dependent preprocessing phase. We rely on the masked secret-sharing protocol from ABY2.0 [33], with preprocessing performed using the HE-based Cheetah protocol [23]. This enables a constant-round preprocessing phase.

Both frameworks are implemented in C++. In both implementations, we use a bit length of 32 bits to represent secret shares, i.e., all arithmetic is performed over the ring $\mathbb{Z}_{2^{32}}$. For reproducibility, we provide our implementation in an anonymised Git repository available at [12]. We deploy our implementation on a testbed and evaluate it on a concrete dataset to characterise the performance of *MPCTM*. We describe the testbed and the dataset below.

a) **Testbed:** For this testbed, we consider two drones equipped with larger computational resources. We use two embedded Intel NUC7i5BNK devices [25], each equipped with an Intel Core i5-7260U processor running at 2.20 GHz, 16GB of RAM, and a 256GB SSD, running Ubuntu 20.04.6 LTS. The *Testbed* emulates drones characterised by more significant computational power, such as high-end drones. For communication, we leverage an ad hoc IEEE 802.11 network operating at the frequency of 2.4 GHz in a regular office environment. This setting is intentional, since it allows our proof-of-concept to share the same spectrum with other communications and possibly experience interference and packet losses, enabling a realistic test environment.

b) **Dataset:** To provide fairness in the comparison, we leverage the dataset used in [19], which is based on real-world trajectories released in [31] and available at [38]. The dataset contains the trajectories of several taxis in the city of Porto, collected between July 2013 and June 2014, sampled every 15 seconds, with varying number of points, ranging from $N = 3$ to $N = 100$. We use the prepared dataset from [19], classified as a collision dataset, which contains 88 trajectories that collide in space but not necessarily over the time axis. In addition, we further process the dataset to parametrise our cube and rectangle/ellipse approaches (see section IV). For the cube representation, we process the trajectories point by point, while for the rectangle/ellipse approach, we pre-compute the parameters for each pair of points representing the rectangle. To validate correctness, we evaluated our implementation on the real-world drone dataset from [17], incorporating its GPS, velocity, and sampling-rate data into our model. Although collected at low altitudes and under conditions intended for RGB image acquisition rather than collision scenarios, the dataset confirms that our implementation operates on real drone data [17].

B. Performance Evaluation

In this section, we describe the experimental setting followed by performance metric of *MPCTM*.

a) **Experiment Settings:** We evaluate *MPCTM* based on four relevant metrics, allowing us to understand deployment feasibility and identify bottlenecks. These metrics are execution time, memory footprint, bandwidth, and energy consumption on both devices. We report the memory footprint, specifically RAM consumption, in the appendix. For execution time, we measure the time taken by the entire protocol until it identifies collisions in space and time, in seconds. In addition, we measure the time taken by each protocol round on both the initiatior (client) and the responder (server) until the output of the final decision about the collision or non-collision status of the considered input trajectories. We report the bandwidth as the application payload generated by the schemes to identify collisions. For *MPCTM*, we use the integrated benchmark reporting from HPMPC and MP-SPDZ, while for the work in [19], we use their implementation to obtain the corresponding results. Finally, we estimate the energy consumption of *MPCTM*. To do so, we rely on the tool *powertop*, a software-based energy estimation tool. The tool *powertop* provides an estimate of the power, in Watt, of a running process. Afterwards, we compute the energy consumption of the scheme by multiplying the power estimate by the execution time, as $E = P \times \Delta t$, where E is the estimated energy in Joule, P is the power, in Watt, estimated by *powertop*, and Δt is the duration of the collision identification phase of *MPCTM*. To compare our work with [19], we use their reported configuration. We configure the collision threshold as $\epsilon = 50$ meters, the GNSS inaccuracy as $\sigma = 50$ meters, and the circle steps of the responder as $\eta = \frac{\pi}{20}$. We report the results of each metric using the Empirical Cumulative Distribution Function (ECDF). For a given stochastic process X, the ECDF reports the empirical probability that X is lower than or equal to a given value x, i.e., X, $ECDF(x) = y$ implies that $p(X \leq x) = y$.

b) **Performance Metric:** In Fig. 4, we illustrate the execution time of all methods we analysed. For our solutions, we distinguish between the *total* time and the *online* time. The total time captures the autonomous UAS-to-UAS system model. In contrast, the *online* time assumes the existence of a semi-honest dealer, e.g., a non-colliding UAS within radio visibility or a ground station, as stated in Sec. III. We discuss this setting in detail in Sec. VI.

We observe that the gray line in Fig. 4 (a), representing the preprocessing plus online phase, and the gray dashed line, representing the online phase only, correspond to the Π_{OBRTM} protocol. This protocol requires the largest execution time in the MP-SPDZ framework. Considering the 0.95 quantile of the ECDF, the online phase requires 62.3 s on both the server and the client. When including the preprocessing phase, the total time increases to 73.20 s.

For the Π_{PointTM} protocol in MP-SPDZ, represented by the pink line and pink dashed line in Fig. 4 (a), the online phase requires 3.6 s, while the total time corresponds to 5.7 s, at the 0.95 quantile, for both the server and the client.

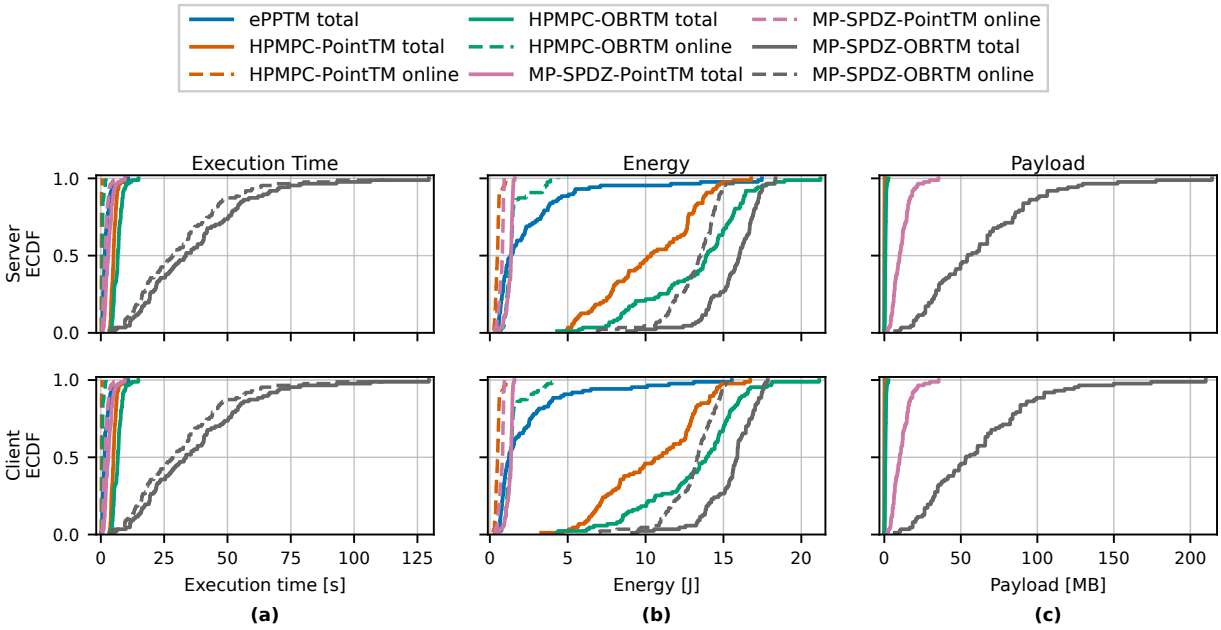


Fig. 4: Performance evaluation of our solution and ePPTM across three metrics: (a) execution time in seconds, (b) energy consumption in Joule, and (c) application payload in Megabytes.

In the HPMPC framework, our Π_{OBRTM} protocol, represented by the green line and green dashed line (Fig. 4(a)), requires 1.73 s for the online phase and 9.75 s of total runtime for both the server and the client. In parallel, the Π_{PointTM} protocol, represented by the orange line and orange dashed line in Fig. 4, requires 0.51 s for the online phase, while the total time is 7.02 s for both the server and the client. Finally, ePPTM, represented by the blue line, does not have a preprocessing phase. Therefore, we report only its total time, which requires 5.28 s on both the server and the client.

We observe a similar trend for the energy consumption, especially for the online phase. The Π_{OBRTM} protocol in MP-SPDZ requires the largest online energy consumption, with 15 J on both devices (dashed gray line). When considering the total energy consumption, this increases to 17.51 J on the server and 17.57 J on the client (gray line). In contrast, the same protocol in HPMPC requires 3.55 J and 3.40 J for the online phase on the server and client, respectively, while the total energy consumption is 17.39 J and 16.69 J.

For the Π_{PointTM} protocol, HPMPC requires 0.89 J and 0.88 J for the online phase on the server and client, respectively. In MP-SPDZ, the online phase requires 0.90 J and 0.91 J on the server and client, respectively. When considering the total energy consumption, HPMPC requires 14.58 J and 14.58 J, while MP-SPDZ requires 1.50 J and 1.48 J on the server and client, respectively. Finally, ePPTM does not have a preprocessing phase. Therefore, we report only its total energy consumption, which is 6.88 J and 8.31 J on the server and client, respectively. Relative to propulsion, this overhead is negligible: at a measured hover power of approximately 300 W [20], the mean 0.493 J consumed by Π_{PointTM} in dealer mode corresponds to roughly 2 ms of hover, and the worst case across all our configurations remains below 60 ms.

Regarding the application payload, we observe that the MP-SPDZ implementations generate the largest amount of exchanged data. The Π_{OBRTM} protocol in MP-SPDZ requires 124.56 MB and 122.23 MB on the server and client, respectively. In contrast, the same protocol in HPMPC requires only 1.55 MB on both devices, showing a significant reduction in communication overhead.

For the Π_{PointTM} protocol, MP-SPDZ requires 21.04 MB and 21.15 MB on the server and client, respectively, while HPMPC requires only 0.34 MB on both devices. Finally, ePPTM requires the smallest application payload, with 0.12 MB on the server and 0.08 MB on the client. However, ePPTM does not distinguish between preprocessing and online phases. Therefore, we report only its total application payload.

Overall, we observe that the HPMPC framework has a clear advantage over the other methods when the preprocessing phase is delegated. In the comparison between HPMPC and MP-SPDZ, the building blocks used in HPMPC accelerate the execution time and reduce the communication overhead. Additionally, we leverage the Single Instruction Multiple Data (SIMD) implementation of HPMPC to parallelise the comparisons in Π_{OBRTM} and Π_{PointTM}.

We report the mean performance of all methods we measured in Table II. We distinguish between the online/dealer mode and the total/autonomous mode. For ePPTM, we report the performance of both the client and the server, since one party has additional overhead due to the computation of intermediate points for collision detection. In contrast, our solution requires no such separation, since both devices

execute equivalent protocol steps symmetrically (see Sec. IV).

Our evaluation highlights the trade-offs between the different protocols and MPC backends. MP-SPDZ benefits from low RAM consumption due to its function-independent preprocessing. This comes at the cost of high runtime overhead, especially for our modelled OBR-based protocol. In contrast, HPMPC demonstrates the advantage of optimised function-dependent preprocessing, although it requires substantially more RAM. HPMPC enables a fast online phase and outperforms ePPTM in online runtime by a factor of $2\times$ with our modelled protocol and $8\times$ with our MPC-friendly protocol.

Both implementations incur higher communication than ePPTM, as expected from the use of MPC. In autonomous mode, our implementations introduce an overhead of approximately $1.5\times$–$3\times$ in total runtime. This overhead comes with end-to-end security and preserves the privacy of the underlying trajectory data, revealing no information beyond the prescribed output, unlike ePPTM.

c) **Accuracy Analysis:** The authors in [19] claim to achieve 100% accuracy in the space domain. However, as described in the experimental setting, we use static configurations that influence their trajectory representation. In particular, their approach does not model the time interval nor the velocity when representing the trajectory. Therefore, their limitations are strongly associated with the chosen dataset and configuration. This leads to false negatives, which jeopardize the safety of UASs.

In contrast, our protocols consider these limitations. We account for the velocity and the time interval, allowing us to identify the actual number of collisions in space and time. Moreover, Π_{PointTM} is efficient in identifying collisions, although it introduces false positives. This is preferable in our setting, since reporting possible collisions is safer than missing actual collisions.

Overall, the $F1$ score demonstrates that Π_{OBRTM} is the most suitable trajectory representation to identify collisions, followed by Π_{PointTM}, which reports a small number of false positives, in contrast to [19] that reports false negatives.

VI. Discussion

a) **Protocol Choice:** The choice between the two protocols is governed by spatial dimension and false-positive tolerance. In three dimensions the separating-axis test requires fifteen candidate axes rather than four, raising the online cost by roughly $3.4\times$. Π_{PointTM} is therefore the only practical choice for aerial and underwater vehicles. In two dimensions, Π_{OBRTM} is preferred whenever false positives are costly, since its endpoint-constrained region is strictly tighter, while Π_{PointTM} remains preferable under tight latency or memory budgets. In both cases the protocol must complete within the trajectory sampling interval. The back-end choice is separate: where a semi-honest dealer is available, HPMPC's function-dependent preprocessing gives the fastest online phase. In autonomous mode, f-independent preprocessing of MP-SPDZ is preferable for Π_{PointTM}, where it improves simultaneously on runtime, energy and memory, whereas Π_{OBRTM} remains faster under HPMPC. The complete selection procedure is summarised by a decision graph in Figure 9 in Appendix E.

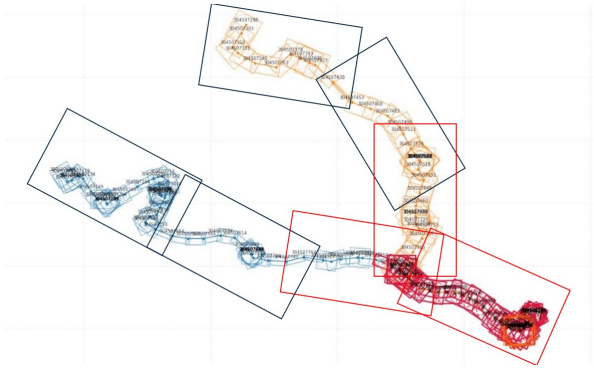


Fig. 5: Illustration of the worst-case scenario among the 88 trajectories, where several larger rectangles still overlap and therefore require further refinement

b) **Performance Overhead:** The reason why our HPMPC-based solutions take longer than ePPTM [19] is mainly due to the *preprocessing* phase. In contrast, if a ground station or another semi-honest dealer is available, our solution outperforms ePPTM [19]. At the same time, ePPTM [19] state that their approach leaks information about the direction of the UAS. This represents a privacy leakage, whereas our solution reveals only the colliding points.

This leads to the next concern, namely the resolution of collisions. After identifying the colliding points, both parties may rearrange their trajectories and run the protocol again. If they use the HPMPC-based protocols with a semi-honest dealer, then the protocol can be rerun efficiently. Otherwise, another execution requires an additional preprocessing phase, increasing the chance that the UASs have already reached the next trajectory point at the given time and velocity. However, collision-resolution strategies are out of scope in this paper, and we leave it as future work.

The 0.95 percentile of Π_{OBRTM} requires approximately 9 s. In the current dataset, the sampling time is 15s, which confirms that, under this sampling configuration, the UAS can identify collisions before reaching the next trajectory point. However, if the sampling time becomes smaller, we have to rely either on the setting with a semi-honest dealer or on Π_{PointTM}, which is more efficient but may introduce false positives.

c) **Optimisation:** A possible solution to the aforementioned limitations is to map multiple consecutive trajectory points to a larger OBR and identify collisions among such larger rectangles, as shown in Figure 5. Figure 5 presents a layer-wise approach, similar to a "cascade", where each rectangle contains multiple trajectory points. In a later stage, only the trajectory pairs represented by colliding larger rectangles (marked as red) are further refined and compared. This may reduce the time of the *preprocessing* phase, since fewer pairs of rectangles need to be compared and, consequently, fewer Beaver triples need to be generated. However, depending on the level of cascading, the preprocessing time can also

TABLE II: Mean performance evaluation of all methods and measured metrics.

Method	Online / Dealer Mode				Total / Autonomous Mode			
	Time (s)	Energy (J)	Payload (MB)	RAM (MB)	Time (s)	Energy (J)	Payload (MB)	RAM (MB)
ePPTM (Server)	2.09	2.55	0.048	124.07	2.09	2.55	0.048	124.07
ePPTM (Client)	1.97	2.33	0.031	123.98	1.97	2.33	0.031	123.98
MP-SPDZ-Π_{OBRTM}	31.40	13.15	1.38	111.65	37.11	15.64	62.09	111.65
MP-SPDZ-Π_{PointTM}	1.83	0.735	0.280	83.03	3.10	1.28	10.98	83.03
HPMPC-Π_{OBRTM}	0.771	1.52	0.566	493.29	6.78	13.20	0.768	493.29
HPMPC-Π_{PointTM}	0.252	0.493	0.115	389.01	5.14	10.25	0.172	389.01

TABLE III: Collision detection performance of the evaluated methods.

Method	TP	TN	FP	FN	Accuracy	F1
ePPTM [19]	68	8	3	9	86.36%	91.89%
MPCTM Π_{PointTM}	77	4	7	0	92.05%	95.65%
MPCTM Π_{OBRTM}	77	11	0	0	100.00%	100.00%

accumulate, since each additional refinement layer requires further comparisons. This cannot be controlled in advance, because both parties do not know a priori which portions of their trajectories are collision-prone without already solving the collision-detection problem.

d) **Scalability:** In this work, we target the two-party setting described in Sec. III. Our current design does not directly extend to collision detection among multiple UASs, unless each UAS runs multiple instances of our protocol with every other UAS within radio visibility. Thus, our design is geometrically bounded to identifying collisions between two trajectory pairs represented as OBRs, as in Π_{OBRTM}, or between trajectory points, as in Π_{PointTM}. A possible future direction is to leverage the semi-honest dealer to coordinate collision avoidance among multiple parties in a privacy-preserving manner. The dealer could reveal only the colliding trajectory points to the involved UASs, without disclosing with whom they may collide or the other parties' trajectories. However, real-time collision avoidance remains challenging when no semi-honest dealer is available. This requires further investigation on how to model multi-party trajectories while revealing only the points involved in an actual collision, rather than broadcasting the collision information to all UASs.

e) **Limitations:** Our work protects private trajectory inputs but necessarily reveals the prescribed collision output. Because each semi-honest party knows its own trajectory, it may combine outputs from repeated executions with prior knowledge to constrain the other party's movement. The resulting leakage depends on the number, ordering, distribution, and coverage of colliding points. Consecutive or geometrically informative collisions may reveal the approximate direction of travel and narrow the feasible intermediate path [19], whereas sparse collisions provide isolated spatio-temporal constraints without uniquely determining the trajectory. Subsequent executions use only points or segments updated by the selected collision-resolution strategy, which may reduce but does not eliminate cumulative leakage.

f) **Open Challenges:** We reported the computational overhead of our solution and, based on the results, we identified two open challenges. First, we observe that the main bottleneck for UASs is the *preprocessing* phase. In the current literature on state-of-the-art MPC, the preprocessing phase is often delegated or outsourced, and the main optimisation effort focuses on the *online* phase. However, in real-time systems, where decisions must be taken immediately while preserving both privacy and safety, the preprocessing phase becomes critical. This requires revisiting preprocessing mechanisms and investigating how to improve their efficiency, as also considered in [4]. This leads to the following open research question: *How can preprocessing be made efficient for real-time systems while minimizing communication rounds and maintaining security against semi-honest or active adversaries?*

Second, guaranteeing input verifiability remains an open challenge for active security. Without input verifiability, a malicious UAS could provide fake trajectory points for collision detection while following a completely different path. Thus, even if the underlying protocol is secure, the correctness of the provided trajectory cannot be guaranteed. This raises the following open research question: *How can privacy-preserving collision detection guarantee that the submitted trajectory points are consistent with the actual motion of the UAS, without revealing the private trajectory itself?*

VII. Conclusion

In this paper, we proposed *MPCTM*, a novel Multi-Party Computation-based scheme for privacy-preserving trajectory matching among UASs in shared operational space. *MPCTM* enables commercial UASs operating under unreliable or scarce internet connectivity to identify collisions without revealing their trajectory points. When a dealer is available, *MPCTM* can further delegate the preprocessing phase to reduce the computational overhead on the UASs. Since trajectory points can reveal the mission or characteristics of an UAS from its navigational pattern, *MPCTM* allows the involved parties to protect their paths and reveal only collision-prone points.

We described *MPCTM* and showed its security against a semi-honest adversary trying to infer the trajectory points of the other party during collision detection. Moreover, we implemented an open-source proof-of-concept and experimentally evaluated its accuracy and overhead in autonomous and dealer mode. Our evaluation shows that *MPCTM* accurately identifies collisions in space and time, while remaining lightweight in

dealer mode. Future work will extend *MPCTM* to multi-UAS collision detection and malicious security.

Acknowledgments

This work is supported by the European Network of Excellence dAIEDGE under Grant Agreement Nr. 101120726 and by the Flemish Government through the Cybersecurity Research Program with grant number: VOEWICS02.

Artifact

We make the complete artifact publicly available at [12]. We provide the full source code of all implementations presented in this work, including the HPMPC- and MP-SPDZ-based realisations of our protocols. We also include detailed documentation with instructions for reproducing all benchmarks reported in the paper.

References

[1] AgNet West. From Backyard Prototype to Global Ag-Tech: Carbon Robotics' LaserWeeder Revolution, 2023. Accessed: 2026-05-04.

[2] BBC News. Drones and business: What is happening in the industry, 2023. Accessed: 2026-05-04.

[3] Donald Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO*, 1991.

[4] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 291–308, New York, NY, USA, 2019. Association for Computing Machinery.

[5] British Geological Survey. How we use drones, 2024. Accessed: 2026-05-04.

[6] Davide Callegaro and Marco Levorato. Optimal edge computing for infrastructure-assisted uav systems. *IEEE Transactions on Vehicular Technology*, 70(2):1782–1792, 2021.

[7] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

[8] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority. In *CRYPTO*, 2018.

[9] Jiahe Cui, Shuyao Shi, Yuze He, Jianwei Niu, Guoliang Xing, and Zhenchao Ouyang. VILAM: Infrastructure-assisted 3d visual localization and mapping for autonomous driving. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1831–1845, Santa Clara, CA, April 2024. USENIX Association.

[10] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *NDSS*, 2015.

[11] Anushka Desai, Oscar G. Bautista, and Kemal Akkaya. Privacy-Preserving Collision Detection for Drone-based Aerial Package Delivery using Secure Multi-Party Computation. In *Int. Symp. on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, MobiHoc '23, page 510–515, 2023.

[12] Dominik Roy George, Martin Zbudila. Open Source Code of the Implementation of MPCTM. https://github.com/DominikRoy/Lightweight-Trajectory-Matching, 2026. Accessed: 2026-11-09.

[13] Drone Remote-ID. Tech specifications for Remote ID range. https://drone-remote-id.com/#tech_specifications_for_remote_id_range, 2023. (Accessed: 2025-Mar-27).

[14] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 823–852. Springer, 2020.

[15] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[16] FlyByGuys. Drones for maritime, 2023. Accessed: 2026-05-04.

[17] Michael Fonder and Marc Van Droogenbroeck. Mid-air: A multi-modal dataset for extremely low altitude drone flights. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, June 2019.

[18] Keith B Frikken and Mikhail J Atallah. Privacy Preserving Route Planning. In *Proc. of ACM Workshop on Privacy in the Electronic Society*, pages 8–15, 2004.

[19] Dominik Roy George and Savio Sciancalepore. ePPTM—Enhanced Privacy-Preserving Trajectory Matching on Autonomous Vehicles. *IEEE Internet of Things Journal*, 12(13):24552–24569, 2025.

[20] Hao Gong, Baoqi Huang, Bing Jia, and Hansu Dai. Modeling power consumptions for multirotor uavs. *IEEE Transactions on Aerospace and Electronic Systems*, 59(6):7409–7422, 2023.

[21] S. Gottschalk. Separating Axis Theorem. Technical Report TR96-024, Department of Computer Science, University of North Carolina at Chapel Hill, 1996.

[22] Christopher Harth-Kitzerow, Ajith Suresh, Yongqin Wang, Hossein Yalame, Georg Carle, and Murali Annavaram. High-Throughput Secure Multiparty Computation with an Honest Majority in Various Network Settings. *Proceedings on Privacy Enhancing Technologies*, 2025(1):250–272, 2025.

[23] Zhicong Huang, Wen jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and Fast Secure Two-Party Deep Neural Network Inference. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 809–826, Boston, MA, August 2022. USENIX Association.

[24] Inside Unmanned Systems. Amsterdam's canals float fleet of autonomous boats for passenger transport, waste collection, environmental sensing, 2023. Accessed: 2026-05-04.

[25] Intel. Intel NUC7i5BNK. https://www.intel.com/content/dam/support/us/en/documents/mini-pcs/nuc-kits/NUC7i5BN_NUC7i7BN_TechProdSpec.pdf, 2019. (Accessed: 2026-May-25).

[26] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending Oblivious Transfers Efficiently. In *CRYPTO*, 2003.

[27] Zhuliang Jia, Suprio Ray, Rongxing Lu, and Pulei Xiong. Plosq3d: Privacy-preserving location-based skyline query over encrypted 3d spatial data. *IEEE Transactions on Cloud Computing*, 2026.

[28] Marcel Keller. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[29] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Paul Whatmough, Aleksandra Faust, Sabrina Neuman, Gu-Yeon Wei, David Brooks, and Vijay Janapa Reddi. Automatic domain-specific soc design for autonomous unmanned aerial vehicles. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 300–317, 2022.

[30] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

[31] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. Predicting Taxi–Passenger Demand Using Streaming Data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402, 2013.

[32] Lewis Nkenyereye, Chi Harold Liu, and JaeSeung Song. Towards Secure and Privacy Preserving Collision Avoidance System in 5G Fog based Internet of Vehicles. *Future Generation Computer Systems*, 95:488–499, 2019.

[33] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. {ABY2. 0}: Improved {mixed-protocol} secure {two-party} computation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2165–2182, 2021.

[34] Václav Pritzl, Matouš Vrba, Yurii Stasinchuk, Vít Krátký, Jiří Horyna, Petr Štěpán, and Martin Saska. Drones guiding drones: Cooperative navigation of a less-equipped micro aerial vehicle in cluttered environments. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10597–10604, 2024.

[35] Savio Sciancalepore and Dominik Roy George. Privacy-Preserving Trajectory Matching on Autonomous Unmanned Aerial Vehicles. In *Annual Computer Security Applications Conference*, pages 1–12, 2022.

[36] Katie Searles. Costa Coffee deploys Starship Technologies delivery robots in Milton Keynes, May 2021. Accessed: 2026-05-04.

[37] Spherical Insights & Consulting. Top 50 Companies in Commercial Drone Market: Statistics Report Till 2035, September 2025. Accessed: 2026-05-04.
[38] Luigi Libero Lucio Starace. Porto Taxi Trajectories — figshare.com. https://figshare.com/articles/dataset/Porto_taxi_trajectories/12302165, 2020. (Accessed 2025-Mar-27).
[39] Jingchao Sun, Rui Zhang, and Yanchao Zhang. Privacy-Preserving Spatiotemporal Matching for Secure Device-to-Device Communications. *IEEE Internet of Things Journal*, 3(6):1048–1060, 2016.
[40] Alisson R. Svaigen, Azzedine Boukerche, Linnyer B. Ruiz, and Antonio A. F. Loureiro. MixDrones: A Mix Zones-Based Location Privacy Protection Mechanism for the Internet of Drones. In *Proc.of the 24th Int. ACM Conf.on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '21, page 181–188, 2021.
[41] P. Tedeschi, S. Sciancalepore, and R. Di Pietro. Lightweight Privacy-Preserving Proximity Discovery for Remotely-Controlled Drones. In *Proc.of the 39th Annual Computer Security Applications Conference*, ACSAC '23, page 178–189, 2023.
[42] Pietro Tedeschi, Siva Ganesh Ganti, and Savio Sciancalepore. Selective Authenticated Pilot Location Disclosure for Remote ID-enabled Drones. In *Proc.of Privacy Enhancing Technology Symposium*, 2024.
[43] Pietro Tedeschi, Savio Sciancalepore, and Roberto Di Pietro. PPCA - Privacy-Preserving Collision Avoidance for Autonomous Unmanned Aerial Vehicles. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2022.
[44] Xu Wei-jiang, Jing Wei-wei, Huang Liu-sheng, and Yao Yi-fei. Privacy-Preserving Collision Detection of Two Circles. In *Int. Conf. on Scalable Information Systems*, InfoScale '07, 2007.
[45] Ayong Ye, Yacheng Li, Li Xu, Qing Li, and Hui Lin. A Trajectory Privacy-Preserving Algorithm Based on Road Networks in Continuous Location-Based Services. In *IEEE Trustcom/BigDataSE/ICESS*, pages 510–516, 2017.
[46] Martin Zbudila, Ajith Suresh, Hossein Yalame, Omid Mirzamohammadi, Aysajan Abidin, and Bart Preneel. SoK: Connecting the Dots in Privacy-Preserving ML - Systematization of MPC Protocols and Conversions Between Secret Sharing Schemes. Cryptology ePrint Archive, Paper 2025/1679, 2025.
[47] Liang Zhang, Haibin Kan, and Yihao Wang. Privacy-Preserving AGV Collision-Resistance at the Edge Using Location-Based Encryption. *IEEE Transactions on Services Computing*, 16(4):2868–2878, 2023.

APPENDIX A
REACHABLE REGION

In Definition A.1 we formally define the elliptic region with foci of two consecutive trajectory points as the reachable region between them. In Definition A.2 we define the OBR encapsulating such elliptic region.

Definition A.1 (Reachable space between two trajectory points)**.** Let P_i and P_{i+1} be two consecutive sampled points of a trajectory, with spatial positions

$$\mathbf{r}_i, \mathbf{r}_{i+1} \in \mathbb{R}^2$$

and timestamps $t_i < t_{i+1}$. Let $v_{\max} > 0$ denote the maximum speed of the vehicle and define

$$\Delta t_i = t_{i+1} - t_i, \qquad \delta_i = v_{\max} \Delta t_i.$$

The reachable region between P_i and P_{i+1} is the ellipse

$$\mathcal{E}_i = \left\{ \mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x} - \mathbf{r}_i\|_2 + \|\mathbf{x} - \mathbf{r}_{i+1}\|_2 \le \delta_i \right\}.$$

Definition A.2 (Oriented bounding rectangle)**.** Let $\mathcal{E}_i$ be the reachable ellipse between two consecutive trajectory points P_i and P_{i+1}. Define its center as

$$\mathbf{m}_i = \frac{\mathbf{r}_i + \mathbf{r}_{i+1}}{2}.$$

Let

$$a_i = \frac{\delta_i}{2}, \qquad b_i = \sqrt{a_i^2 - \left(\frac{\|\mathbf{r}_{i+1} - \mathbf{r}_i\|_2}{2} \right)^2}.$$

The unit vector along the major axis is

$$\mathbf{u}_i = \begin{cases} \dfrac{\mathbf{r}_{i+1} - \mathbf{r}_i}{\|\mathbf{r}_{i+1} - \mathbf{r}_i\|_2}, & \mathbf{r}_{i+1} \neq \mathbf{r}_i, \\ \mathbf{e}_1, & \mathbf{r}_{i+1} = \mathbf{r}_i, \end{cases}$$

and let $\mathbf{v}_i$ be the unit vector orthogonal to $\mathbf{u}_i$. Here, $\mathbf{e}_1$ denotes the first standard basis vector. The oriented bounding rectangle of $\mathcal{E}_i$ is

$$\mathcal{R}_i = \{ \mathbf{m}_i + \alpha \mathbf{u}_i + \beta \mathbf{v}_i : |\alpha| \le a_i,\ |\beta| \le b_i,\ t_i \}.$$

We denote the spatial representation of this rectangle by the tuple

$$\rho_i = (\mathbf{m}_i, \mathbf{u}_i, \mathbf{v}_i, a_i, b_i).$$

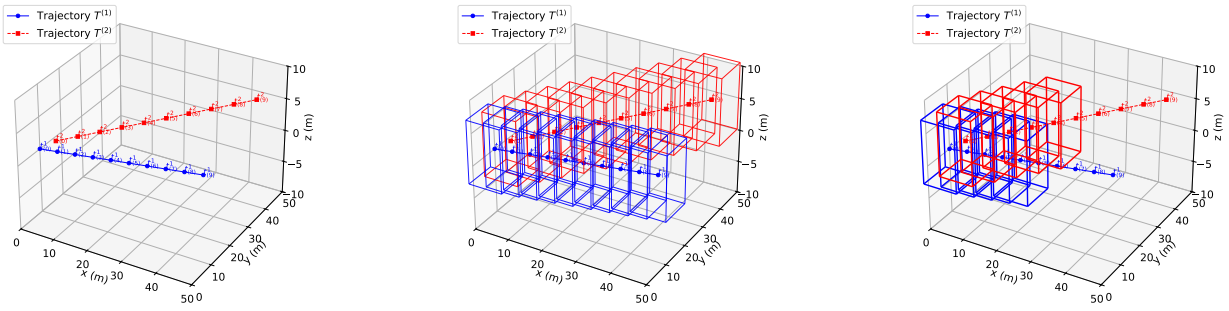
APPENDIX B
CUBE-BASED TRAJECTORY MATCHING

Figure 6 depicts the three-dimensional representation of trajectories and visualises collision detection proposed in Protocol 3.

APPENDIX C
COST ANALYSIS

We express all costs over the ring $\mathbb{Z}_{2^\ell}$ in terms of the primitives of Sec. III, as instantiated by our two backends. Linear operations are local and therefore free. A multiplication and a bit injection each require one reconstruction. An inner product of dimension d requires one reconstruction in the masked-sharing protocol of ABY2.0 [33], whose online communication is independent of the vector dimension, and d reconstructions in the Beaver-triple protocol of Semi2k, which is SPDZ$_{2^k}$ [8] stripped of amplifying, sacrificing, MAC generation and OT correlation checks, leaving the generation of additively shared Beaver triples through OT [28]. In ABY2.0 a Boolean AND gate costs two bits online for any fan-in up to four, which we exploit both in the final conjunction and in the adder used for comparison.

The only back-end-dependent primitive is Π_{LTZ}, realised as a sign-bit extraction, for which we write r_{cmp} and c_{cmp} for its online depth and communication. In HPMPC we use the parallel-prefix adder of ABY2.0, which combines four generate–propagate pairs per prefix node using AND gates of fan-in up to four. Extracting the sign bit of an ℓ-bit value requires at most $(\ell - 2)/3$ such nodes, each evaluating its four gates in a single round, so $r_{\mathsf{cmp}} = \lceil \log_4 \ell \rceil$ and $c_{\mathsf{cmp}} = 8(\ell - 2)/3$ bits. The higher fan-in reduces both depth and online communication, at the price of a larger preprocessing cost per gate. In MP-SPDZ we enable edaBits [14], giving $r_{\mathsf{cmp}} = 1 + \lceil \log_2 \ell \rceil$ and $c_{\mathsf{cmp}} = 2\ell + 8(\ell - 2)$ bits. Base costs are collected in Table IV. Throughout, N and N' denote the number of sampled points of the two UASs, κ is the computational security parameter, and communication is counted in bits summed over both parties.



(a) (b) (c)

Fig. 6: (a) presents the trajectory of $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$. (b) illustrates an example of representing trajectory as point-centric cubes. (c) showcases possible point collisions among the two trajectories highlighted in bold cubes.

TABLE IV: Online cost of the underlying MPC primitives over $\mathbb{Z}_{2^\ell}$. Left: the masked-sharing protocol of ABY2.0 with function-dependent preprocessing instantiated using Cheetah [23] and parallel-prefix sign-bit extraction, as implemented in HPMPC [22]. Right: the additive Beaver-triple protocol Semi2k with function-independent preprocessing and edaBit-based comparison, as implemented in MP-SPDZ [28].

Primitive	ABY2.0 + PPA		Semi2k + edaBits	
	Rounds	Comm.	Rounds	Comm.
Π_{Share} (per element)	1	ℓ	0	0
Linear operation	0	0	0	0
Multiplication	1	2ℓ	1	4ℓ
Inner product, dim. d	1	2ℓ	1	$4d\ell$
Bit injection	1	2ℓ	1	4ℓ
AND (2–4-input)	1	2	1	4
Π_{LTZ}	$\lceil \log_4 \ell \rceil$	$8(\ell-2)/3$	$1+\lceil \log_2 \ell \rceil$	$2\ell+8(\ell-2)$
Π_{Recon} (one bit)	1	2	1	2

a) Communication complexity of Π_{OBRTM}*:* The $N \cdot N'$ rectangle pairs are mutually independent and are evaluated as a single SIMD batch, so the round complexity of Protocol 1 does not depend on the trajectory lengths. Within one pair, Eq. (3) requires only eight distinct inner products, namely $\mathbf{d}\cdot\mathbf{n}$ for the four candidate separating axes and the four frame-to-frame terms $\mathbf{u}^{(1)}\cdot\mathbf{u}^{(2)}$, $\mathbf{u}^{(1)}\cdot\mathbf{v}^{(2)}$, $\mathbf{v}^{(1)}\cdot\mathbf{u}^{(2)}$ and $\mathbf{v}^{(1)}\cdot\mathbf{v}^{(2)}$, since $\mathbf{u}^{(1)}\cdot\mathbf{u}^{(2)} = \mathbf{u}^{(2)}\cdot\mathbf{u}^{(1)}$. These (Steps 8–10) and the squaring of t_{ij} (Step 15) are mutually independent and are evaluated in a single parallel round. Under function-dependent preprocessing the online cost of an inner product is independent of its dimension at unchanged preprocessing cost [33], so each of the eight products costs a single reconstruction, whereas the function-independent instantiation pays two. Each absolute value is realised as $|x| = x - 2b\cdot x$ with $b \leftarrow \Pi_{\mathsf{LTZ}}(x)$, i.e. one comparison followed by one bit injection. The four separating-axis comparisons (Step 12) and the temporal comparison (Step 17) are independent and are merged into a single batched Π_{LTZ} invocation. Then, the five Boolean shares are combined by an AND tree of depth $\lceil \log_2 5 \rceil = 3$ with 2-input gates, or of depth 2 using the multi-input AND gates of ABY2.0 [33]. Input sharing costs one round and ℓ bits per element in the masked-sharing setting, and is entirely non-interactive under function-independent preprocessing, where the sender derives its mask from a pairwise PRNG. Moreover, $\mathbf{v}_i$ is the 90° rotation of $\mathbf{u}_i$ and is obtained by a local linear map, so only seven ring elements per rectangle are shared. Finally, Δt and $(\Delta t)^2$ are computed once and amortised over all pairs. Per pair the protocol therefore evaluates 8 inner products, 9 ring multiplications (8 scalar products of a half-side length with an absolute value, plus the squaring of t_{ij}), 8 bit injections, 13 comparisons and 4 AND gates, and reconstructs a single bit. The operation count is presented in Table V. The online round complexity is $7 + 2\,r_{\mathsf{cmp}}$ in both instantiations: the additional sharing round of the masked-sharing protocol is exactly offset by the shallower AND tree it admits. In $D = 3$ dimensions the separating-axis test requires $K = 15$ axes instead of $K = 4$, and the counts generalise to $K + D^2$ inner products and absolute values, $KD + 1$ multiplications, $2K + D^2 + 1$ comparisons and K AND gates.

b) Communication complexity of Π_{PointTM}*:* The MPC-friendly protocol replaces the geometric predicate by $D + 1$ coordinate-wise squared comparisons, where D denotes the number of spatial dimensions. Per pair of sampled points, Protocol 3 evaluates $D + 1$ squarings (Step 6), all of which are independent and are batched into one round, followed by $D+1$ comparisons (Step 8) that are likewise independent and batched into a single Π_{LTZ} invocation. The $D + 1$ Boolean shares are combined by an AND tree of depth $\lceil \log_2(D+1) \rceil$, which collapses to a single 4-input AND gate for $D = 3$ under

TABLE V: Operation count of Π_{OBRTM} and Π_{PointTM} per pair of trajectory elements $(i, j) \in [N] \times [N']$, together with the one-off terms that are amortised over all pairs. K denotes the number of candidate separating axes and D the number of spatial dimensions ($K = 4$ for $D = 2$, $K = 15$ for $D = 3$).

Primitive	Π_{OBRTM} (D=2)	Π_{PointTM}	general
Per pair ($N \cdot N'$)			
Inner product (dim. D)	8	–	$K + D^2$
Multiplication	9	D+1	$KD + 1$
Bit injection	8	–	$K + D^2$
Π_{LTZ}	13	D+1	$2K + D^2 + 1$
AND	4	D	K
Π_{Recon} (one bit)	1	1	1
One-off (independent of $N \cdot N'$)			
Π_{Share} (ring elements)	$7(N+N')+2$	$(D+1)(N+N')+4$	–
Δt / Π_{Bounds}	1 max, 1 sq.	2 sq.	–

TABLE VI: Concrete online cost and preprocessing material per pair of trajectory elements, for $\ell = 32$ and $D = 3$, using the primitive costs of Tab. IV. Preprocessing is reported as the number of correlated-randomness objects. Instantiated with OT extension [26] a ring product costs $2\ell(\kappa+\ell)$ bits, whereas the RLWE-based generation of Cheetah [23] reduces this by roughly two orders of magnitude to the cost of $\mathcal{O}(\ell)$.

Protocol	Backend	Online rounds	Online comm. [bit]	Preprocessing material
Π_{OBRTM}	ABY2.0 + PPA	$7 + 2r_{\mathsf{cmp}} = 13$	$50\ell + 13c_{\mathsf{cmp}} + 6 = 2\,646$	25 OLE, 8 bit-inj.
	Semi2k + edaBits	$7 + 2r_{\mathsf{cmp}} = 19$	$132\ell + 13c_{\mathsf{cmp}} + 18 = 8\,194$	25 triples, 8 daBit, 13 edaBit(ℓ)
Π_{PointTM}	ABY2.0 + PPA	$4 + r_{\mathsf{cmp}} = 7$	$8\ell + 4c_{\mathsf{cmp}} + 4 = 580$	4 OLE
	Semi2k + edaBits	$4 + r_{\mathsf{cmp}} = 10$	$16\ell + 4c_{\mathsf{cmp}} + 14 = 1\,742$	4 triples, 4 edaBit(ℓ)

ABY2.0, and one bit is reconstructed. The threshold derivation Π_{Bounds} contributes two squarings. It is independent of the pairwise loop, is evaluated once, and its $\mathcal{O}(1)$ cost is amortised over all $N \cdot N'$ pairs. As above, sharing is non-interactive under function-independent preprocessing and costs one round and $\ell(D+1)(N+N')+4\ell$ bits otherwise. The resulting online round complexity is $4+r_{\mathsf{cmp}}$, again identical for both backends, and the per-pair online communication is $8\ell + 4c_{\mathsf{cmp}} + 4$ bits under function-dependent and $16\ell + 4c_{\mathsf{cmp}} + 14$ bits under function-independent preprocessing for $D = 3$. Compared to Π_{OBRTM} this is a reduction by a factor of $4.56\times$ and $4.70\times$ respectively, against the payload ratios of $4.92\times$ and $4.93\times$ measured in Table II. The concrete costs are presented in Table VI.

c) Deviations of the implementation: The analysis above describes an idealised evaluation and consequently underestimates the measured ratios by 8% and 5%. Three implementation choices account for the gap, none of which affects the round complexity of the pairwise loop or the asymptotic behaviour. The final conjunction and the output reconstruction operate on arithmetic rather than Boolean shares, so each costs a ring element instead of a bit, and the prototype instantiates Π_{LTZ} with the two-input prefix adder rather than the four-input variant assumed above. Second, the two protocols use opposite SIMD layouts: Π_{OBRTM} packs four independent rectangle pairs per register, whereas Π_{PointTM} places the four coordinates of a single pair in the four lanes and therefore does not amortise its last two rounds over several pairs, which inflates its measured payload relative to the model and thus the observed ratio. Closing these gaps is a matter of engineering rather than of protocol design.

d) Preprocessing complexity: Function-independent preprocessing generates each Beaver triple through OT extension at $2\ell(\kappa + \ell)$ bits [26], so that in our measurements the preprocessing payload exceeds the online payload by $44\times$ for Π_{OBRTM} and $38\times$ for Π_{PointTM}. Function-dependent preprocessing amortises a ring product to $\mathcal{O}(\ell)$ bits using the RLWE-based generation of Cheetah [23], keeping the preprocessing payload *below* the online one ($0.36\times$ and $0.50\times$) and roughly two orders of magnitude smaller in absolute terms. The price is storage: the correlated randomness is bound to specific wires of a specific circuit, so all $N \cdot N'$ instances must be generated and retained until the online phase begins, rather than drawn from a generic pool as needed. This is directly visible in the memory footprint of Table II, where the function-dependent backend consumes $4.4\times$ and $4.7\times$ more RAM than the function-independent one. Function-dependent preprocessing therefore reduces both online and total communication, but its memory requirement bounds the number of trajectory pairs a resource-constrained UAS can preprocess in a single batch.

APPENDIX D
SECURITY ANALYSIS

We analyse security in the Universal Composability (UC) framework [7] against the static semi-honest adversary $\mathcal{A}$ of Sec. III-E, which corrupts at most one of P_1, P_2 in the autonomous model, and at most one of P_1, P_2, D in the dealer-

assisted model, where D denotes the dealer and collusion between D and a computing party is excluded.

Figures 7 and 8 define the ideal functionalities $\mathcal{F}_{\mathsf{OBRTM}}$ and $\mathcal{F}_{\mathsf{PointTM}}$ realised by Protocols 1 and 3. Both receive the encoded trajectories of the two parties, evaluate the collision predicate pairwise, and return the resulting collision bits to both parties. Both also receive a temporal bound from each party, but the two bounds carry different meanings and are combined differently, since the two representations bound time differently. In $\mathcal{F}_{\mathsf{OBRTM}}$ a rectangle covers the entire interval between two consecutive samples, and that interval has length at most $t^{(j)}_{\mathsf{max}}$. Two rectangles can therefore overlap in time only if $|t^{(1)}_i - t^{(2)}_j| \le \max\big(t^{(1)}_{\mathsf{max}}, t^{(2)}_{\mathsf{max}}\big)$, so the single global bound Δt never discards a genuine overlap and no per-pair quantity is required. In $\mathcal{F}_{\mathsf{PointTM}}$ each sampled point instead carries the symmetric tolerance $t^{(j)}_i \pm t^{(j)}$, where $t^{(j)}$ is *half* the largest sampling interval, so that the windows of consecutive points tile the timeline without overlapping. Two such windows intersect exactly when $|t^{(1)}_i - t^{(2)}_j| \le t^{(1)} + t^{(2)}$, and the threshold is therefore a sum. A maximum would miss collisions there, whereas a sum in $\mathcal{F}_{\mathsf{OBRTM}}$, or full rather than half intervals in $\mathcal{F}_{\mathsf{PointTM}}$, would be sound but needlessly loose. Their remaining input interfaces differ, mirroring the protocols they abstract. $\mathcal{F}_{\mathsf{PointTM}}$ additionally receives the distance $d^{(j)} := v^{(j)}_{\mathsf{max}} t^{(j)}$ that each party can travel within its half-interval, from which it derives the squared spatial thresholds of Protocol 2. In $\mathcal{F}_{\mathsf{OBRTM}}$ no spatial threshold is required, since the geometry of each oriented bounding rectangle already accounts for the party's own maximum velocity when it constructs $\rho^{(j)}_i$ (Definition A.2). Both take fixed-point encoded inputs (Definition III.1) and evaluate exactly the collision predicate of Sec. IV. The accuracy discussion of Sec. V therefore concerns the functionality itself and is orthogonal to security.

The only information leaked beyond the prescribed output is the pair (N_1, N_2) of trajectory lengths, which is exactly the leakage granted to the dealer in Sec. III-D. We stress that this quantity is not meaningful on its own. Since *MPCTM* is sampling-agnostic, each party selects its sampling strategy and rate independently and never discloses it (Sec. IV). A given N_j is therefore consistent with any travelled distance and any mission duration, and cannot be mapped to a spatial or temporal extent. This is in contrast to schemes that fix a common sampling rate in advance, where the length of the input vector directly bounds the distance travelled by the other party and must be hidden through padding.

Both protocols are described in the $\mathcal{F}_{\mathsf{ABB}}$-hybrid model, where $\mathcal{F}_{\mathsf{ABB}}$ is the standard arithmetic black box over $\mathbb{Z}_{2^\ell}$ that stores values under handles $\langle\cdot\rangle$, $\langle\cdot\rangle^B$ and exposes the commands Share, Lin, Mult, LTZ, AND and Recon of Sec. III-C. Only Recon returns a value to the parties, all other commands return handles. We instantiate $\mathcal{F}_{\mathsf{ABB}}$ with the masked-sharing protocol of ABY2.0 [33] using Cheetah [23] preprocessing, and with the semi-honest variant of SPDZ$_{2^k}$ [8] using OT-based preprocessing [26]. Both UC-realise $\mathcal{F}_{\mathsf{ABB}}$

Functionality $\mathcal{F}_{\mathsf{OBRTM}}$

Parametrised by the ring $\mathbb{Z}_{2^\ell}$. It interacts with P_1, P_2 and, in dealer mode, with a dealer D that provides no input and receives no output.

Input On input $\big(\mathcal{R}^{(j)}, t^{(j)}_{\mathsf{max}}\big)$ from P_j, $j \in \{1,2\}$, with $\mathcal{R}^{(j)} = \big(\rho^{(j)}_i, t^{(j)}_i\big)_{i=1}^{N_j}$ and $\rho^{(j)}_i = \big(\mathbf{m}^{(j)}_i, \mathbf{u}^{(j)}_i, \mathbf{v}^{(j)}_i, a^{(j)}_i, b^{(j)}_i\big)$, where $t^{(j)}_{\mathsf{max}}$ is P_j's largest sampling interval, record the input and send (len, j, N_j) to $\mathcal{A}$.

Compute Once both inputs are recorded, set $\Delta t := \max\big(t^{(1)}_{\mathsf{max}}, t^{(2)}_{\mathsf{max}}\big)$. For every $(i,j) \in [N_1] \times [N_2]$ let $\mathbf{d}_{ij} := \mathbf{m}^{(2)}_j - \mathbf{m}^{(1)}_i$ and, for every $\mathbf{n} \in \{\mathbf{u}^{(1)}_i, \mathbf{v}^{(1)}_i, \mathbf{u}^{(2)}_j, \mathbf{v}^{(2)}_j\}$, let $r^{(1)}_i(\mathbf{n}), r^{(2)}_j(\mathbf{n})$ be the projection radii of Eq. (1). Define

$$\Delta_{\mathbf{n}ij} := |\mathbf{d}_{ij} \cdot \mathbf{n}| - r^{(1)}_i(\mathbf{n}) - r^{(2)}_j(\mathbf{n}),$$
$$\Delta t_{ij} := \big(t^{(1)}_i - t^{(2)}_j\big)^2 - \Delta t^2,$$

and compute

$$s_{ij} := \textstyle\bigwedge_{\mathbf{n}} \mathbf{1}\big(\Delta_{\mathbf{n}ij} < 0\big) \wedge \mathbf{1}\big(\Delta t_{ij} < 0\big).$$

Output Send $\{s_{ij}\}_{(i,j)\in[N_1]\times[N_2]}$ to P_1 and P_2.

Fig. 7: Functionality $\mathcal{F}_{\mathsf{OBRTM}}$.

Functionality $\mathcal{F}_{\mathsf{PointTM}}$

Parametrised by the ring $\mathbb{Z}_{2^\ell}$, the coordinate set $\mathcal{C} = \{x, y, z, t\}$ ($\mathcal{C} = \{x, y, t\}$ in the two-dimensional setting), and a public margin ϵ. It interacts with P_1, P_2 and, in dealer mode, with a dealer D that provides no input and receives no output.

Input On input $\big(\mathcal{T}^{(j)}, d^{(j)}, t^{(j)}\big)$ from P_j, $j \in \{1,2\}$, with $\mathcal{T}^{(j)} = \big(\mathbf{r}^{(j)}_i, t^{(j)}_i\big)_{i=1}^{N_j}$, where $t^{(j)}$ is half of P_j's largest sampling interval and $d^{(j)} := v^{(j)}_{\mathsf{max}} t^{(j)}$ the distance P_j can travel within it, record the input and send (len, j, N_j) to $\mathcal{A}$.

Bounds Once both inputs are recorded, set $d_{\mathsf{total}} := d^{(1)} + d^{(2)} + \epsilon$ and $t_{\mathsf{total}} := t^{(1)} + t^{(2)}$. Define $\Delta d_n := d_{\mathsf{total}}$ for every $n \in \mathcal{C} \setminus \{t\}$ and $\Delta d_t := t_{\mathsf{total}}$.

Compute For every $(i,j) \in [N_1] \times [N_2]$ and every $n \in \mathcal{C}$ define $\Delta_{nij} := \big(n^{(1)}_i - n^{(2)}_j\big)^2 - \Delta^2 d_n$, and compute

$$s_{ij} := \textstyle\bigwedge_{n\in\mathcal{C}} \mathbf{1}\big(\Delta_{nij} < 0\big).$$

Output Send $\{s_{ij}\}_{(i,j)\in[N_1]\times[N_2]}$ to P_1 and P_2.

Fig. 8: Functionality $\mathcal{F}_{\mathsf{PointTM}}$.

against a static semi-honest adversary corrupting one party. In dealer mode the preprocessing is replaced by a correlated-randomness functionality $\mathcal{F}_{\mathsf{CR}}$ that takes no input from P_1, P_2 and distributes shares of Beaver triples, so the dealer's view is by construction independent of the trajectories.

Theorem D.1. *In the $\mathcal{F}_{\mathsf{ABB}}$-hybrid model, Protocol 1 UC-realises $\mathcal{F}_{\mathsf{OBRTM}}$ and Protocol 3 UC-realises $\mathcal{F}_{\mathsf{PointTM}}$ against*

a static semi-honest adversary corrupting at most one of P_1, P_2, and, in dealer mode, at most one of P_1, P_2, D under the non-collusion assumption of Sec. III-E.

Proof sketch. Both protocols have the same shape: (i) each party shares its own input, (ii) a fixed, input-independent sequence of Lin, Mult, LTZ and AND calls is issued on handles, and (iii) a single Recon is performed on the handles carrying $\{s_{ij}\}$. The threshold derivations are subsumed by (ii): in Protocol 2 the sums $d^{(1)} + d^{(2)} + \epsilon$ and $t^{(1)} + t^{(2)}$ are linear operations and the two squarings are Mult calls, while Steps 1–4 of Protocol 1 contribute one secure maximum and one squaring. Both preambles are evaluated through $\mathcal{F}_{\mathsf{ABB}}$ without reconstruction, and once per execution rather than per pair of trajectory points, so they contribute $O(1)$ to the overall cost.

Let P_1 be corrupted, w.l.o.g. The simulator $\mathcal{S}$ runs $\mathcal{A}$ internally and emulates $\mathcal{F}_{\mathsf{ABB}}$ towards it. It extracts $\mathcal{A}$'s input from its Share calls, submits it to $\mathcal{F}_{\mathsf{OBRTM}}$ (resp. $\mathcal{F}_{\mathsf{PointTM}}$) and receives $\{s_{ij}\}$. All calls of type (i) and (ii) are answered with handles exactly as $\mathcal{F}_{\mathsf{ABB}}$ would, which requires no knowledge of the honest party's input. The single Recon call on $\langle s_{ij}\rangle^B$ is answered with the bit s_{ij} obtained from the ideal functionality. The emulated view is distributed identically to the real $\mathcal{F}_{\mathsf{ABB}}$-hybrid view, and the number of Share calls reveals only (N_1, N_2), which the functionality leaks. Real and ideal executions are therefore perfectly indistinguishable.

If instead $\mathcal{A}$ corrupts D, its entire view consists of the correlated randomness it generates, whose amount is a function of (N_1, N_2) and whose distribution is independent of the parties' inputs. $\mathcal{S}$ samples it with the correct distribution. Non-collusion is required because the dealer's randomness together with the state of a computing party would reconstruct the honest party's shares. Composing with UC-secure instantiations of $\mathcal{F}_{\mathsf{ABB}}$ and $\mathcal{F}_{\mathsf{CR}}$ yields security of the concrete protocols. □

a) Output Leakage.: The output $\{s_{ij}\}$ discloses *which* index pairs collide, and hence a bounded part of the other party's trajectory whenever a collision occurs. This leakage is inherent to the functionality rather than to its realisation, and it is confined to the collision-prone portion of the path: because sampling remains private, an index i by itself carries no spatial or temporal meaning. A variant that reveals only whether a collision exists is obtained by evaluating $\bigvee_{i,j}\langle s_{ij}\rangle^B$ before the final Recon, and is secure by the same argument. We do not claim security against malicious deviations such as malformed inputs, false trajectory submissions or selective aborts, these are discussed in Sec. VI.

Appendix E
Protocol Choice

The protocol and preprocessing choices discussed in the main text can be expressed as the deployment decision tree in Figure 9. The tree first selects the geometric protocol according to spatial dimension and false-positive tolerance. It then selects the preprocessing regime according to dealer availability and, for Π_{OBRTM}, the platform's ability to retain correlated randomness for all $N \cdot N'$ protocol instances.

Appendix F
RAM Consumption

The memory footprint is collected by measuring the maximum resident set size of each process as the RAM consumption of the protocols, in MB. To measure this consumption, we use the Linux tool */usr/bin/time -v* and extract the field *Maximum resident set size*. We present the results in Section F.

Regarding RAM consumption, we observe that the HPMPC implementations require the largest amount of memory. The Π_{OBRTM} protocol in HPMPC requires 621.72MB and 494.02MB on the server and client, respectively. In contrast, the same protocol in MP-SPDZ requires only 121.02 MB and 120.93 MB, indicating that MP-SPDZ has a substantially smaller memory footprint for this protocol.

For the Π_{PointTM} protocol, HPMPC requires 415.23MB and 414.04MB on the server and client, respectively, while MP-SPDZ requires only 97.74MB and 98.90MB. Finally, ePPTM has a stable memory footprint of 125.34MB on the server and 125.22MB on the client. Since ePPTM does not distinguish between preprocessing and online phases, we report only its total RAM consumption.

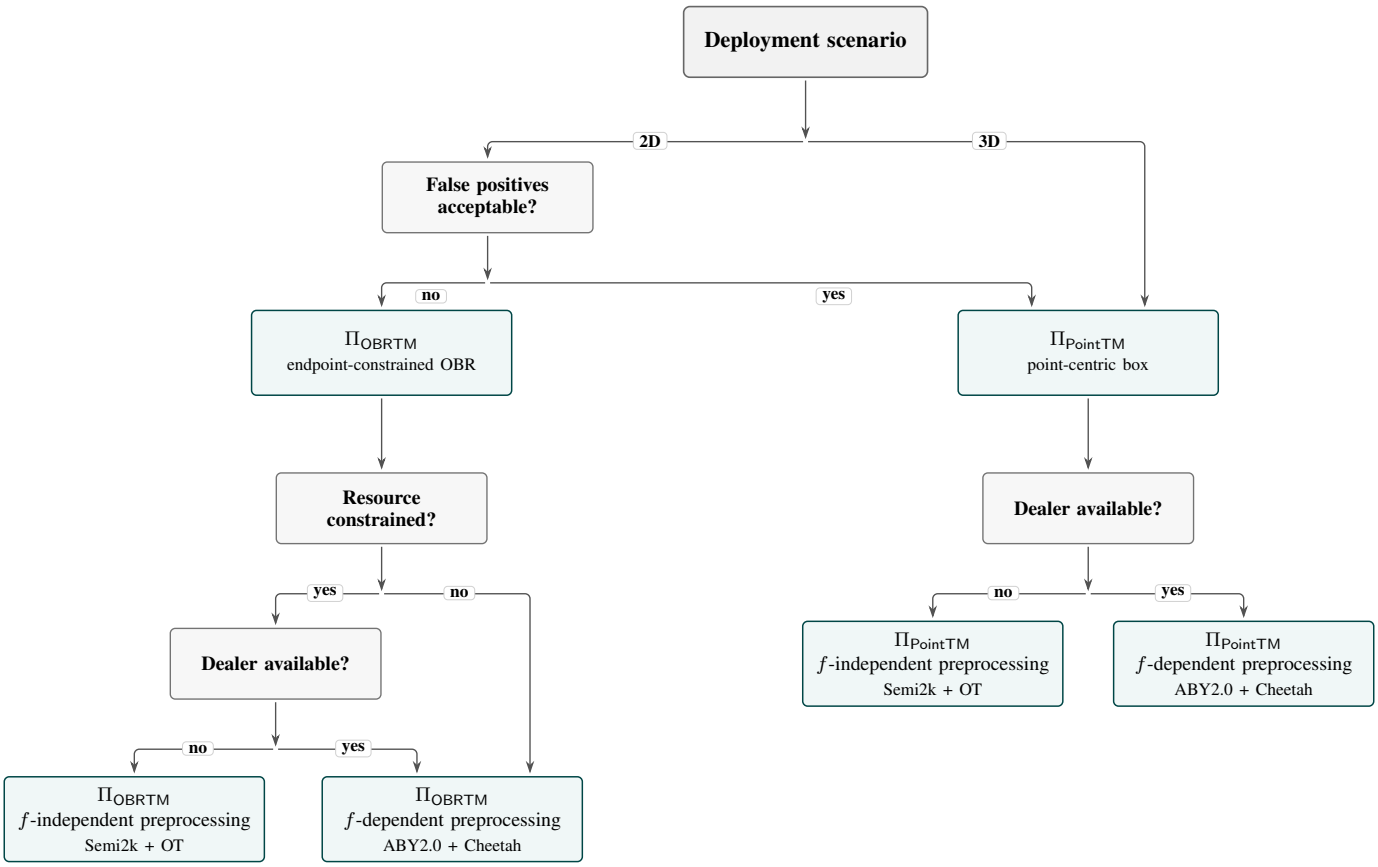


Fig. 9: Deployment decision tree for *MPCTM*. Π_{OBRTM} takes function-dependent preprocessing unless the platform cannot retain the correlated randomness for all $N \cdot N'$ instances and no dealer is available. For Π_{PointTM} the choice turns on dealer availability alone, since in autonomous mode function-independent preprocessing is preferable on runtime, energy and memory simultaneously (Table II).

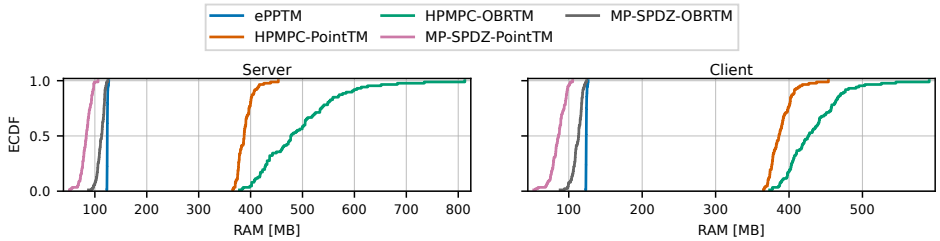


Fig. 10: Performance evaluation of our solution and ePPTM in terms of memory footprint.