Cyclops: Binding a Vehicle's Digital Identity to its Physical Trajectory using Monocular Cameras

Lewis W. Koplon*, Ameer G. Nessaee*, Alex Choi*, Andress Mentoza†, Michael Villasana*, Loukas Lazos*, and Ming Li*
*Electrical and Computer Engineering, University of Arizona
†Engineering Physics and Electrical Engineering, New Mexico State University

***Abstract*—We address the problem of cyber-physical access control for connected autonomous vehicles. The goal is to bind a vehicle's digital identity to its physical identity represented by its physical properties such as its trajectory. We highlight that simply complementing digital authentication with sensing information remains insecure. A remote adversary with valid or compromised cryptographic credentials can hijack the physical identities of nearby vehicles detected by sensors. We propose a cyber-physical challenge-response protocol named *Cyclops* that relies on low-cost monocular cameras to perform cyber and physical identity binding. In *Cyclops*, a verifier vehicle challenges a prover vehicle to prove its claimed physical trajectory. The prover constructs a response by capturing a series of scenes in the common Field of View (cFoV) between the prover and the verifier. Verification is achieved by matching the dynamic targets in the cFoV (other vehicles crossing the cFoV). The security of *Cyclops* relies on the spatiotemporal traffic randomness that cannot be predicted by a remote adversary. We validate the security of *Cyclops* via simulations on the CARLA simulator and on-road real-world experiments in an urban setting.**

***Index Terms*—cyber-physical trust, V2V communications, authentication, trajectory verification.**

I. Introduction

Connected autonomous vehicles (CAVs) fuse information collected from onboard sensors and vehicle-to-everything (V2X) communications to perform crucial autonomy operations such as cooperative perception [1], path planning [2], platooning [3], [4], collision avoidance [5], and others. Sensing modalities such as Radar/LiDAR and cameras are typically limited to line-of-sight (LoS) and have sampling rates constrained by the hardware/software stack. V2X messages complement sensory information by communicating the physical state (location/velocity/acceleration) of nearby agents, as well as their intent of motion (future acceleration, steering, and trajectory). For instance, cooperative adaptive cruise controllers (CACCs) deployed in autonomous platooning applications can reduce the platooning distance from 3 seconds to as low as 0.5 seconds at highway speeds, thus increasing a platoon's fuel efficiency and improving the traffic density without compromising safety [6].

The reliance on over-the-air data for decision-making opens CAVs to new vulnerabilities. False data injections could lead to deadly accidents and heavy monetary losses due to decreased traffic efficiency and cargo loss [7], [8]. To combat such threats, wireless standards such as the IEEE 1609.2 [9] and the more recent 3GPP TS 33.185 for Cellular Vehicle-to-Everything (C-V2X) [10] recommend the use of a public key infrastructure (PKI). However, digital authentication does not validate the physical identity of a message originator. As a result, remote adversaries with valid or compromised cryptographic credentials can inject false V2X messages without being physically present in the environment.

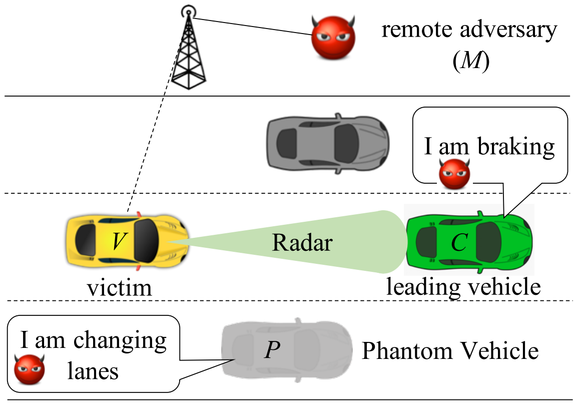


Fig. 1: A remote attack without cyber-physical identity binding. The adversary claims the existence of a phantom vehicle P and sends false messages to a victim vehicle. Alternatively, the adversary hijacks the physical identity of $\mathcal{C}$ which is sensed by $\mathcal{V}$.

An example of this attack is shown in Fig. 1. A remote attacker $\mathcal{M}$ embeds a phantom vehicle P with public key pk_P and certificate $cert_P$, into $\mathcal{V}$'s environment. Phantom vehicle P advertises a physical state tuple $(p_P(t), \vec{v}_P(t), \vec{a}_P(t))$, representing P's position, velocity, and acceleration at time t. Moreover, P claims the intent to switch to $\mathcal{V}$'s lane. Note that $\mathcal{V}$ may need to react to P's claim before sensors can validate P's existence. In a more advanced attack, the remote adversary hijacks the identity of an existing vehicle C that is leading $\mathcal{V}$. The attacker claims to have a digital identity $pk_P, cert_P$ (which can be cryptographically validated) but hijacks $\mathcal{C}$'s physical trajectory by advertising tuple $(p_C(t), \vec{v}_C(t), \vec{a}_C(t))$. We emphasize that a physical-identity hijacking attack cannot be prevented by sensing, since a vehicle with $(p_C(t), \vec{v}_C(t), \vec{a}_C(t))$ can be sensed. Moreover, remote attacks can scale to many locations, as the adversary is not bound to be physically co-present.

Symposium on Vehicles Security and Privacy (VehicleSec) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-7-6
https://dx.doi.org/10.14722/vehiclesec.2024.23052
www.ndss-symposium.org

Some known primitives such as distance bounding [11] and proximity verification [12], [13] can be integrated into security protocols to provide cyber-physical access control. However, they come with significant limitations. First, they only verify location, leaving other physical properties such as velocity, acceleration, and trajectory unauthenticated. Moreover, distance bounding and proximity verification are crude estimators and do not allow for inferring information such as lane and relative positioning. In many instances [14]–[16], they are limited to small distances that are not usable for vehicular applications. Finally, they may require custom hardware and a custom PHY layer to remain secure [17].

More recent solutions that focus on vehicular applications aim at verifying the physical properties of vehicles [18]–[21]. These include secure localization and tracking [19] or motion verification approaches [21], [22] which validate the data consistency between a vehicle's claim and environment data (e.g., RF signal characteristics). However, verification is still crude. In the context of platooning applications, physical access control is applied during the vehicle admission phase [22]–[26]. To be admitted into a platoon, a prover must first prove that she follows the platoon at the following distance. The *Convoy* protocol proposed by Han *et al.* uses the vertical acceleration due to road surface variations to correlate the prover's and the verifier's trajectories [23]. However, an adversary can pre-record the vertical acceleration caused by road surface variations ahead of time, making *Convoy* vulnerable to pre-recording attacks. Xu *et al.* mitigated pre-recording attacks by proposing a proof-of-following (PoF) protocol that exploits large-scale fading correlation in ambient LTE signals to establish physical proximity. Both the *Convoy* and the PoF protocols cannot establish the relative vehicle ordering, exact distance, or trajectory. Dickey *et al.* added relative ordering and lane verification by proposing the *Wiggle* protocol. In *Wiggle*, the prover is challenged to perform a series of perturbations of her physical trajectory to validate her trajectory claim. Such perturbations are not always possible depending on the traffic flow and may impact the user experience.

To address the limitations of prior methods, we propose *Cyclops*, a cyber-physical challenge-response (CPCR) protocol that relies on the vision modality to bind a vehicle's digital identity with its physical one. Specifically, we make the following contributions:

- We develop *Cyclops*, a CPCR protocol executed between a prover and a verifier vehicle to thwart remote attacks. *Cyclops* draws security from the spatiotemporal randomness in nearby traffic to verify the prover's claimed trajectory. The prover and the verifier synchronously capture scenes in their common field-of-view (cFoV) using low-cost monocular cameras. Matching objects in the cFoV are used to validate the prover's claimed trajectory.
- We propose a fuzzy scene-matching algorithm that accounts for the practical uncertainties in the camera's extrinsic properties and limitations of object detection and depth estimation to generate a security advantage for a truthful prover compared to a remote adversary.
- We implement *Cyclops* in the CARLA open-source vehicle simulator to define the protocol's security parameters. We further conduct real-world experiments in an urban setting to demonstrate the security of *Cyclops* to remote attacks and highlight practical challenges. The code for implementing the core primitive of *Cyclops* is available at the following repository [27].

II. System and Threat Models

System Model. We consider the interaction between two vehicles via V2V messages. One vehicle acts as the verifier $\mathcal{V}$ whereas the other acts as the prover $\mathcal{P}$. The prover and the verifier have digital IDs $ID_P{:}(pk_P, cert_P)$ and $ID_V{:}(pk_V, cert_V)$, respectively where pk_X is the public key of $X = (\mathcal{P}, \mathcal{V})$ from a public/private key pair (pk_X, sk_X), and $cert_X$ is the key certificate of X. Such primitives are already defined in vehicular standards (e.g., 3GPP TS 33.185 [10]). Moreover, each party has a physical identity represented by a vector $L_X = (\ell_X(t_1), \ell_X(t_2), \dots, \ell_X(t_n))$, which indicates X's physical trajectory in time. Note that X's physical identity is not static but evolves with time as a vehicle moves through space. In this work, we focus on a vehicle's trajectory as the main physical identity representation. Generally, other attributes can be attached such as vehicle class, color, license plate, velocity, acceleration, and lane. The main motivation for using ephemeral properties to represent the physical identity is because those properties are relevant to the autonomous navigation of the verifier relative to the prover. Moreover, once the physical identity of the prover has been linked to the digital identity, the prover can be tracked via sensors.

Camera Model. Both the prover and the verifier are equipped with monocular cameras. The cameras have intrinsic and extrinsic properties described by the parameter set $C_X = \left(f_X^{(x)}, f_X^{(y)}, z_X, \ell_X, \psi_X, \omega_X, \phi_X\right)$, where $f_X^{(x)}$ and $f_X^{(y)}$ are the camera's horizontal and vertical focal lengths, z_X is the sensor size, $\vec{\ell}_P(i) = (x_X, y_X)$ are the real-world camera coordinates, and ψ_X, ω_X, and ϕ_X are the camera's pitch, roll, and yaw, respectively. The vehicles have enough processing power to execute off-the-shelf object detection algorithms such as YOLO [28] to analyze camera frames and identify objects (vehicles, pedestrians, buildings, etc.). Advancing computer vision algorithms is beyond the scope of this work.

Threat Model. We consider a remote adversary $\mathcal{M}$ who communicates with the verifier through a C-V2X interface or a long-range V2V interface (e.g., by following afar). The adversary has a digital ID, $ID_M = (pk_M, cert_M)$ and claims a physical identity represented by trajectory L_M. The goal of $\mathcal{M}$ is to convince the verifier that she follows L_M, which is related to $\mathcal{V}$'s trajectory L_V. The adversary can impersonate a phantom vehicle or hijack the physical identity of an existing vehicle in the vicinity of $\mathcal{V}$. Finally, the adversary is aware of the statistical model that governs the traffic flow around the verifier. Direct knowledge of detailed traffic model statistics captures the best-case scenario for a remote adversary.

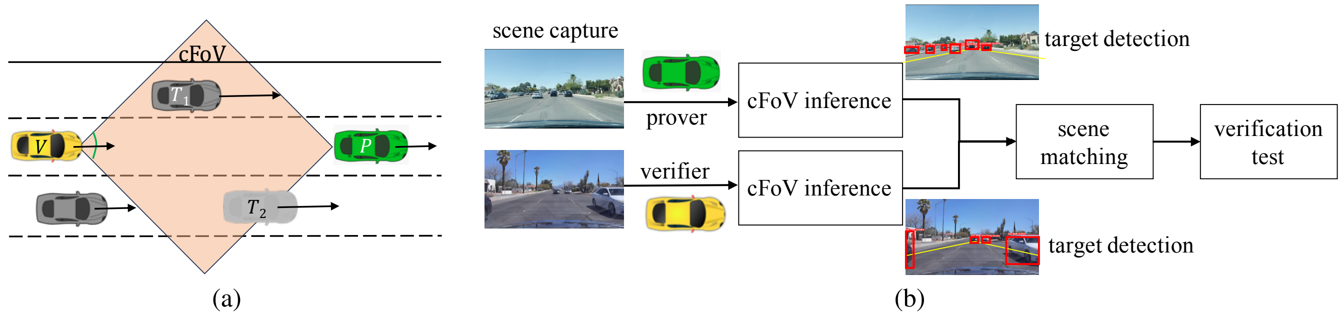


Fig. 2: (a) The verifier challenges the prover to identify targets in the cFoV and (b) the four phases of the *Eye* primitive.

III. The Eye Primitive

A. Overview

To bind the digital identity of a prover vehicle $\mathcal{P}$ with $\mathcal{P}$'s physical identity, we rely on dynamic spatiotemporal environment features extracted through the visual modality. We call the main primitive extracting randomness from traffic, the *Eye*. The main idea of the *Eye* is shown in Fig. 2(a). The *Eye* resembles a game of "I Spy" where the verifier $\mathcal{V}$ challenges the prover $\mathcal{P}$ to identify random features (e.g., objects) within the common surrounding environment traveled by $\mathcal{V}$ and $\mathcal{P}$. Whereas object detection can be implemented using various modalities (e.g., LiDAR, Radar, multi-camera systems), we adopt a low-cost solution using only a single monocular camera per vehicle.

The *Eye* primitive consists of the four phases shown in Fig. 2(b). Initially, the prover claims to follow a trajectory relative to the verifier. The prover's claim is digitally signed, allowing the authentication of $\mathcal{P}$'s digital identity. The verifier challenges the prover to steer its camera to a desired direction where $\mathcal{P}$ and $\mathcal{V}$ can observe cFoV. The prover and the verifier synchronously capture scenes in their FoV. The prover provides the captured images to the verifier as proof of her physical identity. In the next phase and for a given pair of scenes captured by each party, the verifier computes the cFoV in the image domain. Subsequently, the verifier applies object detection to identify targets in the cFoV. Finally, in the scene-matching phase, the verifier matches the targets detected in the prover's scenes to those in his scenes. A proof-of-trajectory test is performed by matching scene pairs over time.

The security of the *Eye* is anchored on the spatial and temporal randomness of traffic, which typically consists of other vehicles passing by in adjacent lanes. A remote adversary is unable to construct scenes with objects in the cFoV at the right time. We now describe the phases of the *Eye* primitive in detail. We focus on the case where the prover leads the verifier on the same lane, but the same approach can be applied to other vehicle configurations, as long as the prover and the verifier have a cFoV. Such configurations are the most relevant for vehicle safety, as V2V messages from vehicles near the verifier directly impact the verifier's control algorithms.

B. Scene-Capture Phase

In the scene-capture phase, $\mathcal{V}$ challenges $\mathcal{P}$ to capture a set of scenes in the $\mathcal{V}$'s FoV to prove her claimed trajectory. The challenge consists of a 4-tuple $\langle t_0, T, s, \phi_P\rangle$ where t_0 is the start time of scene capturing, T is the duration, s is the scene sampling rate, and ϕ_P is the camera yaw for the prover's horizontal FoV. Moreover, the challenge is timestamped to allow for loose synchronization between $\mathcal{V}$ and $\mathcal{P}$. The angle offset ϕ_P is used to control the cFoV between the prover and the verifier, whereas the sampling rate s is used to control the frequency of scene capturing. A carefully tuned sampling rate will reduce the computational overhead of verification while also providing scenes with lower correlation.

Let the number of scenes captured over T be $n = sT$. The prover and the verifier collect sets

$$\begin{aligned} S_P &= \{s_P(1), s_P(2), \ldots, s_P(n)\}, \\ S_V &= \{s_V(1), s_V(2), \ldots, s_V(n)\}, \end{aligned}$$

respectively. Each scene $s_P(i)$ consists of the image frame $im_P(i)$, a timestamp $t(i)$, and the camera's properties represented by C_P. Upon the completion of the scene capture phase, $\mathcal{P}$ sends S_P to $\mathcal{V}$.

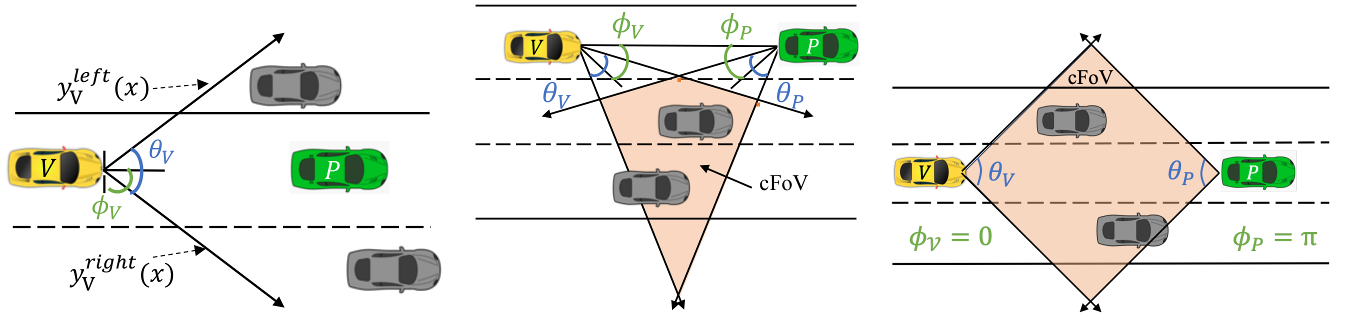
C. Common Field of View Inference Phase

We describe the remaining phases for two scenes $s_P(i)$ and $s_V(i)$, captured at time $t(i)$ and therefore, drop the index i from our notation. The same analysis applies to all scene pairs. Given two scenes s_P and s_V, the verifier determines the cFoV as the intersection between the FoVs of $\mathcal{P}$ and $\mathcal{V}$. The individual FoVs are modeled using real-world boundary lines projected onto the ground plane. Let $y_V^{left}(x), y_V^{right}(x)$ represent the left and right boundary lines of $\mathcal{V}$'s FoV. These lines are illustrated in Fig. 3(a) and expressed by

$$y_V^{left}(x) = \frac{\sin(\phi_V + \frac{\theta_V}{2})}{\cos(\phi_V + \frac{\theta_V}{2})}(x - x_V) + y_V,$$

$$y_V^{right}(x) = \frac{\sin(\phi_V - \frac{\theta_V}{2})}{\cos(\phi_V + \frac{\theta_V}{2})}(x - x_V) + y_V.$$

The angular FoV (aFoV) θ_V represents the observable area through the camera and is calculated by $\theta_V = 2\tan^{-1}\left({z_V}/{2f_V^{(x)}}\right)$. Similarly, the verifier describes the FoV



(a) (b) (c)

Fig. 3: Calculation of the cFoV using their cameras' intrinsic and extrinsic properties. (a) Real-world FoV of $\mathcal{V}$, (b) cFoV with a camera yaw of ϕ_V and ϕ_P and, (c) cFoV when $\phi_V = 0$ and $\phi_P = \pi$ (line-of-sight).

in the prover's scene s_P using lines $y_P^{left}(x)$ and $y_P^{right}(x)$. The verifier computes the real-world polygon $\Pi_{world} = \{\pi_{1,world}, ..., \pi_{4,world}\}$ defined by the intersection of the two FoVs. Two examples of the cFoV polygon are shown in Fig. 3. In Fig. 3(b), we show the cFoV for general camera yaw values ϕ_P and ϕ_V. In Fig. 3(c), we demonstrate the special case where $\mathcal{P}$'s and $\mathcal{V}$'s cameras are aligned to the LoS path ($\phi_V = 0$ and $\phi_P = 180°$).

In the next step, the verifier translates the polygon coordinates in the image domain in both images im_V and im_P obtained from scenes s_V and s_P, respectively. To transform each vertex from real-world coordinates to camera coordinates, the verifier applies the extrinsic camera matrix. Then, the intrinsic matrix transforms the camera coordinates into image coordinates. A detailed explanation of the transformation from the real-world coordinates to the image coordinates is given in [29]. At the end of this process, the verifier has projected the cFoV polygon vertices in Π_{world} onto the image frames in scenes s_P and s_V.

D. Scene-Matching Phase

In the scene-matching phase, $\mathcal{V}$ matches targets in the cFoV between $\mathcal{P}$ and $\mathcal{V}$. Scene matching is a two-step process. In the first step, objects are detected on each image frame. In the second step, the objects detected in scene s_P are translated into the image coordinate system of scene s_V.

Object Detection. To identify targets on a scene s, we applied a pre-trained YOLOv5 model which was tuned for vehicle detection [28], [30]. YOLOv5 performs classification and bounding-box regression, identifying, and boxing any vehicles captured in the input frames. For a scene s, each detected vehicle is assigned a label, stored in the set $\mathcal{O} = \{o_1, o_2, \ldots, o_k\}$. The label for the i^{th} object is defined as $o_i : \{c_i, (u_i, v_i), w_i, h_i\}$, where c_i represents the vehicle class, (u_i, v_i) represents the pixel coordinates of the bounding box center, w_i represents the bounding box width (in pixels), and h_i represents the height (see Fig. 4). Upon completion of this phase, the verifier obtains two sets of objects $\mathcal{O}_P$ and $\mathcal{O}_V$ corresponding to scenes s_P and s_V.

Target Matching. To match the targets in $\mathcal{O}_P$ and $\mathcal{O}_V$, the verifier can apply a perspective transformation to translate

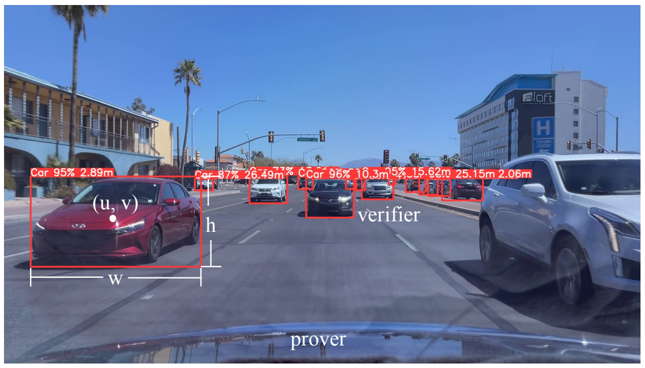


Fig. 4: Detecting objects in an image frame using YOLOv5.

objects to the same coordinate system [31]. To achieve this, a bounding box in the prover's image domain is converted to real-world coordinates using an inverse projection matrix based on the camera's intrinsic and extrinsic parameters. The real-world coordinates are re-projected onto the ego vehicle's image plane using $\mathcal{V}$'s camera parameters. Applying a perspective transformation comes with many challenges in practice. The transformation accuracy is sensitive to the precise knowledge of the camera's extrinsic properties which vary with the camera's mounting stability, road surface, and GPS precision. Moreover, an accurate transformation requires accurate depth estimation, which is poor with monocular cameras.

Region Partitioning. To address these challenges, we propose a fuzzy scene-matching algorithm. The main idea is demonstrated in Fig. 5. The cFoV is partitioned into regions which are mapped to the image domain of s_P and s_V. The bounding box corresponding to each object in $\mathcal{O}_P$ ($\mathcal{O}_V$) is placed in the corresponding region in s_P (s_V). Each scene is then encoded using a binary vector, where the bit b_i representing region R_i is set to 1 if a bounding box is identified to belong in R_i. Scene matching is performed by computing a correlation metric between the respective binary vectors.

To partition the cFoV into regions, the verifier divides it to the left and right sides of the road. The second step involves the horizontal segmentation of the real-world cFoV. Specifically, the verifier identifies ground points that partition the

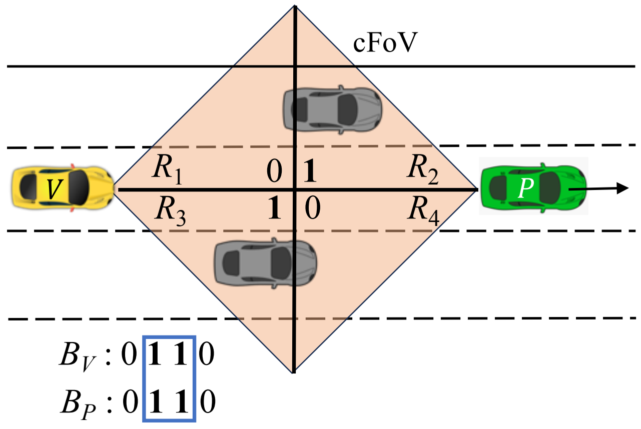


Fig. 5: Partitioning the cFoV to regions and scene encoding.

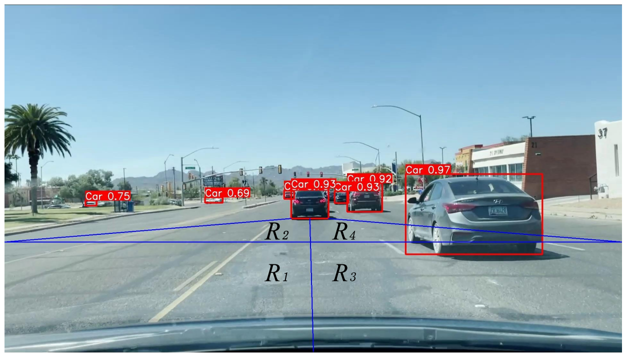


Fig. 6: The cFoV is partitioned into four regions.

verifier-to-prover line into equidistant segments. The ground points are projected to the camera's image domain (similar to the process for the polygon projection). Assuming a level alignment between the camera and the road, a horizontal line through the x pixel coordinate of ground points represents the cFoV region partition in the image domain. Figure 5 shows the partition of the cFoV into four regions R_1, R_2, R_3 and R_4. Fig. 6 shows the partition of the cFoV to four regions in the image domain. Although the regions are equal in the real world, they appear unequal in the image domain due to the image perspective.

Scene Encoding and Matching. To assign an object o_i to a region, we utilize *the bottom edge* in the bounding box representing o_i. We preferred the bottom edge as it was found to be the most accurate compared to other choices such as the center of the bounding box or the intersection-over-union (IoU) area. As illustrated in Fig. 6, the center of the bounding box and the IoU metric would fail to correctly localize the target on the right lane. Each scene is encoded to a bit sequence B of length M where bit b_i corresponds to region R_i. If at least one vehicle is detected at R_i, the corresponding bit is set to one. Otherwise, b_i is set to zero. At the end of this process, the verifier has constructed two bit sequences B_P and B_V based on the prover's and the verifier's scenes s_P and s_V, respectively. Scene matching is performed by comparing the *sequence IoU* against a minimum similarity threshold $\tau \leq 1$. If the IoU exceeds τ, the scenes are considered to match.

$$\gamma = \mathbf{I}\left(\frac{\sum_i \mathbf{I}(b_V(i) = b_P(i) \wedge b_V(i) \neq 0)}{\sum_i \mathbf{I}(b_V(i) \neq 0 \vee b_P(i) \neq 0)} \geq \tau\right). \quad (1)$$

The similarity threshold τ can be less than one to account for cases of vehicle misdetection that inevitably occur on region boundaries, or due to the inaccuracy in the knowledge of the extrinsic camera properties. We study the selection of τ in the evaluation section. For the example in Fig. 5, the verifier computes $B_V = 0110$ and $B_P = 0110$, leading to $\gamma = 1$.

In our context, we prefer the IoU metric over the Hamming distance to deal with the low sequence entropy under sparse traffic conditions. In such conditions, most scenes are empty (without targets), which is reflected with zeros in the scene's bit sequence encoding. An adversary familiar with traffic pattern statistics could achieve a low Hamming distance by simply guessing an all-zero scene. The use of the IoU metric, on the other hand, focuses on vehicle presence, offering a more secure measure of similarity. Because it excludes scenes without vehicles, it forces the adversary to guess the location (region) and time (scene) that nearby vehicles are in the cFoV making it much harder to guess the right sequence of scenes. Note that the IoU metric excludes all empty scenes (no vehicle presence in the cFoV).

IV. The Cyclops Protocol

The *Cyclops* protocol incorporates the *Eye* primitive to bind the prover's digital identity to her physical identity. We describe the protocol's phases in detail.

A. Digital Identity Verification Phase

In this initial phase, the prover claims a trajectory $L_P = (\ell_X(t_1), \ell_X(t_2), \ldots, \ell_X(t_n))$ via a V2V message. The trajectory could be represented by an explicit series of locations or by a state vector $\langle \ell_P(t_1), \vec{v}_P, \vec{a}_P \rangle$ which indicates the current location $\ell_P(t_1)$, velocity $\vec{v}_P$, and acceleration $\vec{a}_P$.

1) For a trajectory L_P, the prover chooses a random nonce r_P and computes $m_1 \leftarrow sig_{sk_P}(ID_V, r_P, L_p)$. The prover sends ID_P, L_P, r_P, and m_1 to the verifier.
2) The verifier checks if $ver_{pk_P}(ID_V, r_P, L_p, m_1) \overset{?}{=} true$. If so, $\mathcal{V}$ "accepts". Otherwise, $\mathcal{V}$ "rejects".

B. Eye Primitive Phase

The verifier challenges the prover by executing the *Eye* primitive on a series of scenes captured over the trajectory L_P. Specifically, the following steps are executed.

5) The verifier chooses the start time of scene capturing t_0, the scene capture duration T, the sampling rate s, and the prover's angle offset (camera yaw) ϕ_P for $\mathcal{P}$'s horizontal FoV. The verifier computes $m_2 \leftarrow enc_{pk_P}(t_0, T, s, \phi_P)$ and $m_3 \leftarrow sig_{sk_V}(ID_P, m_2)$ The verifier sends ID_V, m_2, m_3 to the prover.
6) The prover authenticates $\mathcal{V}$ by checking if $ver_{sk_V}(ID_V, m_2, m_3) \overset{?}{=} true$. The prover decrypts m_2 and captures scenes $S_P = \{s_P(1), s_P(2), \ldots, s_P(n)\}$.

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
Given:		
$ID_P, \langle pk_P, sk_P \rangle, cert_P, pk_V, ID_V, pk_{CA}$		$ID_V, \langle pk_V, sk_V \rangle, pk_P, cert_V, pk_{CA}$
Digital Identity Verification:		
$m_1 \leftarrow sig_{sk_P}(ID_V, r_P, L_p)$		
	$\xrightarrow{ID_P, r_P, L_P, m_1}$	Verify: $ver_{pk_P}(ID_V, r_P, L_p, m_1) \overset{?}{=} true$
***Eye* Primitive:**		
		$m_2 \leftarrow enc_{pk_P}(t_0, T, s, \phi_P)$
		$m_3 \leftarrow sig_{sk_V}(ID_P, m_2)$
Verify: $ver_{pk_V}(ID_V, m_2, m_3) \overset{?}{=} true.$	$\xleftarrow{ID_V, m_2, m_3}$	
Decrypt: $t_0, T, s, \phi_P = dec_{sk_P}(m_2)$		
Capture: $S_P = \{s_P(1), s_P(2), \ldots, s_P(n)\}$		Capture: $S_V = \{s_V(1), s_V(2), \ldots, s_V(n)\}$
$m_4 \leftarrow enc_{pk_V}(S_P)$		
$m_5 \leftarrow sig_{sk_P}(ID_V, m_4)$		
Physical Identity Verification:		
	$\xrightarrow{ID_P, m_4, m_5}$	Verify: $ver_{pk_P}(ID_V, m_4, m_5) \overset{?}{=} true$
		Decrypt: $S_P = dec_{sk_V}(m_4)$
		Compute Γ from S_P and S_V
		Verify: $\mathbf{I}\left(\frac{\left(\sum_{i=1}^{K} \mathbf{I}(H(w_i)/k \geq \alpha)\right)}{K} \geq \beta\right) \overset{?}{=} true$

Fig. 7: The *Cyclops* protocol.

The prover computes $m_4 \leftarrow E_{pk_V}(S_P)$ and $m_5 \leftarrow sig_{sk_P}(ID_V, m_4)$. The prover sends ID_P, m_4, and m_5 to the verifier.

7) The verifier validates the signature on m_4 by checking $ver_{pk_P}(ID_V, m_4, m_5) \overset{?}{=} true$ which verifies that S_P was provided by $\mathcal{P}$, therefore binding ID_P with S_P.

With the completion of the scene capture phase, the verifier has received S_P and collected S_V using his own camera.

C. Physical Identity Verification Phase

During this physical identity verification phase, $\mathcal{V}$ validates that $\mathcal{P}$ follows the claimed trajectory by evaluating S_P.

8) The verifier applies the cFoV and scene-matching phases of the *Eye* primitive on all scene pairs $(s_P(i), s_V(i)), i = 1..n$ and obtains a binary vector Γ of length n. In Γ, $\gamma_i = 1$ if the i^{th} scene pair passes the target matching test in eq. (1) and zero otherwise.
9) The verifier partitions bit sequence Γ into $K = n/k$ words of k bits each. Each word corresponds to k scene pairs. The verifier executes an independent test on each word by checking if the Hamming weight of each word (normalized to the word length) is larger than a threshold α. Finally, the verifier "accepts" if β fraction of word tests are passed.

$$\mathbf{I}\left(\frac{\left(\sum_{i=1}^{K} \mathbf{I}\left(H(w_i)/k \geq \alpha\right)\right)}{K} \geq \beta\right) \overset{?}{=} true \qquad (2)$$

In eq. (2), $H(w_i)$ denotes the Hamming weight of the i^{th} word.

An example of applying the physical identity verification test on Γ is shown below. Sequence Γ is divided into K words $w_1, w_2, \ldots, w_K$ of size k, and the Hamming weight test is applied to each word. If a fraction β of Hamming weights exceeds α, the prover passes verification.

$$\Gamma = \underbrace{\underbrace{\gamma_1, \gamma_2, \ldots, \gamma_k}_{H(w_1)/k \geq \alpha}, \ldots, \underbrace{\gamma_{N-k}, \gamma_{n-k+1}, \ldots, \gamma_n}_{H(w_K)/k \geq \alpha}}_{\geq \beta}.$$

Partitioning the n scenes into K groups and performing K tests (as opposed to a single test on all n scenes) allows the verifier to further decorrelate scenes post collection. This is because correlated scenes will be grouped under a smaller set of words (ideally a single word) thus affecting a single or limited number of tests. In the evaluation, we study the test performance under various combinations of k, α, and β to fine-tune the test parameters. The information flow diagram of Cyclops is shown in Fig. 7.

V. Security Analysis

A. Remote Adversary

To pass the verification of the *Cyclops* protocol, a remote adversary $\mathcal{M}$ must pass the digital and physical identity verifications. Since the adversary holds valid cryptographic credentials, digital identity verification is passed by presenting message m_1 signed with pk_M. To pass physical identity verification, $\mathcal{M}$ must present a scene vector S_M that matches the scene vector S_V collected by the verifier. Because the adversary is remote, she cannot capture scenes in the FoV of the verifier and, therefore, she has to guess those scenes.

Modeling $\mathcal{M}$'s Guessing Strategy. We model the adversary's guessing strategy for S_M as a finite weighted random walk over a Markov chain representing the known traffic distribution in $\mathcal{V}$'s environment. Specifically, let the cFoV be partitioned into $|R|$ regions as detailed in Section III-D. Each scene is encoded to a binary vector B of length $|R|$, indicating the presence or absence of a target in each region. We let the $2^{|R|}$ candidate scenes represent $2^{|R|}$ states in a Markov chain with a probability transition matrix $\mathbf{P}$. Matrix $\mathbf{P}$ captures the probability of transitioning from state s_x to state s_y in the next time step, given the traffic model. The time step is determined by the scene collection sampling rate s.

Generally, the Markovian property does not hold, as a transition from the current state to another depends on prior states. For instance, a vehicle present in region r_i of one lane is more likely to transition to region r_{i+1} of the same lane if a transition from r_{i-1} to r_i happened in a recent time step. This is because the previous recent transition indicates a vehicle traveling at a higher relative velocity leading to correlated transitions. However, if the number of regions is relatively small and the time step is large enough, then a Markovian assumption can be made where the next scene (state) only depends on the previous scene. Under this assumption, the scene transition can be represented as an $|R|$-dimensional Markov chain $G(V, E)$. Figure 8 shows the digraph for the cFoV partition to the left and right lanes (2 regions).

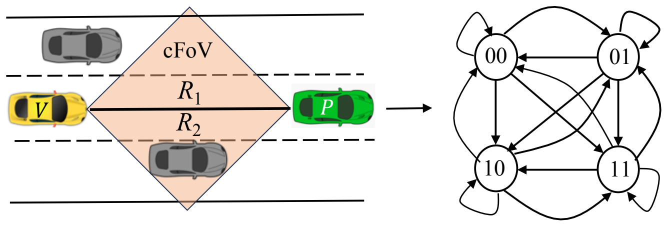


Fig. 8: A 4-state Markov chain representing a cFoV partition to two regions. Transitions to all scenes including self-transitions are possible with probability that is dictated by the traffic model.

Knowing $G(V, E)$, the strategy of the adversary is to decide on a finite walk W_M of length n that maximizes the number of collisions between W_M and the weighted random walk W_V of the verifier. A collision occurs when both walks are in the same state (scene) at the same time step. The total number of collisions in n steps is simply

$$C(n) = \sum_{i=1}^{n} I\left(w_V(i) = w_M(i)\right).$$

where $w_V(i)$ and $w_M(i)$ are the states of W_V and W_M after the i^{th} step. To pass verification, the minimum number of collisions between W_M and W_V must meet the physical identity verification test in Step 9 of the *Cyclops* protocol.

$$C(n) \geq (\beta K)(\alpha k). \tag{3}$$

This is because to pass a single test, the prover must successfully guess (collide) in αk scenes, and a total of βK tests of k scenes each must be passed. Note that passing $(\beta K)(\alpha k)$ tests does not guarantee passing verification because the arrangement of those tests in words is important. The relationship in eq. (3) provides a lower bound. To maximize the collision probability, the adversary can follow a path that starts from the most probable state and transitions to the state with the highest transition probability. However, this strategy can lead to scene patterns that can be easily categorized as adversarial. For instance, if the highest transition probability from a given state is the self-transition probability, the scene pattern stops in a terminating state.

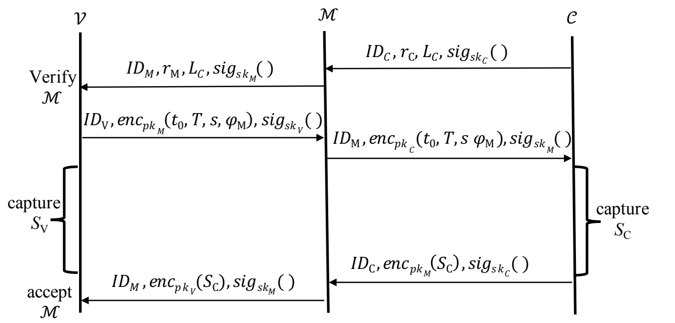

$\mathcal{V}$
$\mathcal{M}$
$\mathcal{C}$
$ID_C, r_C, L_C, sig_{sk_C}()$
Verify $\mathcal{M}$
$ID_M, r_M, L_C, sig_{sk_M}()$
$ID_V, enc_{pk_M}(t_0, T, s, \varphi_M), sig_{sk_V}()$
$ID_M, enc_{pk_C}(t_0, T, s\ \varphi_M), sig_{sk_M}()$
capture S_V
capture S_C
$ID_C, enc_{pk_M}(S_C), sig_{sk_C}()$
accept $\mathcal{M}$
$ID_M, enc_{pk_V}(S_C), sig_{sk_M}()$

Fig. 9: A MitM attack against the *Cyclops* protocol.

A stealthier strategy is to generate a weighted random walk W_M that follows the probability distribution of each state. Such a walk will resemble W_V without having deterministic patterns. To find $C(n)$ under a weighted random walk model, one could apply combinatorial analysis, but this method does not scale with the number of dimensions [32], [33]. Some results are known for infinite graphs (e.g., walks on $\mathbb{Z}^d$ with uniform probability) [32], but our walk model is finite and follows a general distribution over a finite state graph. Intuitively, the number of collisions is inverse to the number of dimensions and the entropy of the distribution. We evaluate $C(n)$ in simulation and real-world experimentation and use parameters α and β to achieve the desired security levels.

B. Man-in-the-Middle Adversary

The adversary $\mathcal{M}$ could also attempt to act as a man-in-the-middle (MitM) between $\mathcal{V}$ and another vehicle $\mathcal{C}$ near $\mathcal{V}$, with the intent of hijacking $\mathcal{C}$'s physical identity. By hijacking another vehicle's physical identity, the adversary can then inject false navigation messages (e.g., intent to brake, change lanes) and impact the control decision-making at the verifier. The MitM attack can be launched as shown in Fig. 9. The adversary receives a trajectory claim L_C from $\mathcal{C}$ and uses the same trajectory claim to $\mathcal{V}$. The verifier validates $\mathcal{M}$'s digital identity and challenges $\mathcal{M}$ to capture scenes starting at time t_0, using ϕ_M. The adversary decrypts the challenge and constructs a new challenge for $\mathcal{C}$ with the same t_0, T, s, and ϕ_M. The verifier and $\mathcal{C}$ synchronously capture scenes in their respective FoVs. The $\mathcal{C}$ responds to the challenge with $enc_{pk_M}(S_C)$. The adversary decrypts to recover S_C and constructs a new response for $\mathcal{V}$ by signing it using her own private key. The verifier performs the physical identity verification test between S_V and S_C. As $\mathcal{C}$ follows the claimed L_C, the adversary passes the test and successfully hijacks the physical identity of $\mathcal{C}$.

The MitM attack is preventable if the prover's and verifier's identities are known to each other. In this closed membership scenario, $\mathcal{C}$ specifies messages for $\mathcal{V}$ and vice versa. The adversary cannot generate a valid signature to challenge $\mathcal{C}$. Moreover, the response from $\mathcal{C}$ is encrypted with $\mathcal{V}$'s public key, and therefore $\mathcal{M}$ cannot obtain S_C. In an open membership scenario where the verifier is not specified, the attack is feasible. This vulnerability can be addressed by incorporating a commitment scheme with a delayed opening phase that renders the scenes relayed from $\mathcal{C}$ to $\mathcal{M}$ stale. The key idea is to force the prover to commit to the captured scenes without revealing them to the verifier. This commitment is then opened with a delay to perform verification. A MiTM adversary spoofing a verifier cannot open the commitment of $\mathcal{C}$ due to the hiding property. The delayed opening phase forces $\mathcal{M}$ to commit to $\mathcal{V}$ without knowing S_C, or commit late with scenes that are not aligned with those of $\mathcal{V}$. Due to space limitations, we omit the details of such protocol.

VI. Evaluation

In this section, we evaluate the correctness and soundness of the *Cyclops* protocol. We use the CARLA Open Urban Driving Simulator [34] to determine the test parameters and then verify the security of *Cyclops* in a real-world setting. The protocol parameters are shown in Table I.

TABLE I: Parameters of the *Cyclops* protocol.

Notation	Definition
$\lvert R\rvert$	Number of regions per scene
τ	Scene similarity threshold
k	Number of scenes per word
α	Normalized Hamming weight threshold for word comparison
K	Number of words tests in the *Cyclops* protocol
β	Minimum fraction of word tests to pass verification

A. Cyclops Parameter Selection

CARLA Simulation Setup. CARLA is an open-source simulator tailored for autonomous vehicle research. The platform supports a flexible sensor suite and full control of all static and dynamic actors. We deployed two vehicles designated as $\mathcal{V}$ and $\mathcal{P}$ which traveled in the middle lane of a highway. Both vehicles were equipped with monocular cameras with a FoV of 105 degrees. The prover led the verifier with cameras oriented to the LoS similar to the topology in Fig. 3(c).

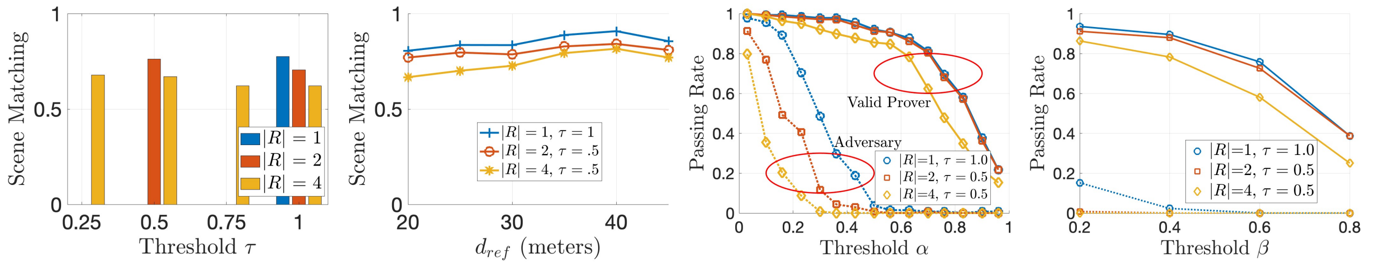
Traffic in each lane was generated according to two traffic models. In the first model termed the *Uniform Traffic Model*, an event consisting of up to two vehicles spawning in the left lane and up to two vehicles spawning in the right lane with uniform probability. Upon triggering the vehicle spawning, each target vehicle was given a throttle to pass through the cFoV thereby simulating the natural flow of traffic. We captured 10 minutes of events across seven discrete distances d_{ref} between the prover and the verifier. The sampling rate was set to 10 fps, culminating in a total of 6,000 scenes per distance. The second model termed the *Urban Traffic Model* was developed from data collected during our on-road experiments in an urban environment. We used the scenes collected during experiments and calculated the traffic distribution (Markov model). We generated scenes according to this distribution for 10 minutes and captured scenes for each $\mathcal{P}-\mathcal{V}$ distance.

B. Cyclops Protocol Parameter Selection

Evaluation of Scene Matching. We first evaluated the accuracy of the scene-matching algorithm when the cFoV is divided into 1, 2, and 4 regions under the *Uniform Traffic Model*. Of all the scenes captured, we excluded scene pairs where both scenes were empty. Figure 10(a) shows the scene matching rate as a function of τ. The threshold for a single region was set to $\tau = 1$, for two regions to $\tau = 0.5, 1$, and for four regions to $\tau = 0.25, 0.5, 0.75$, and 1. The highest matching rate is achieved for $|R| = 1$ because the detection of any number of vehicles within the cFoV is sufficient for scene matching. As the number of regions increases, the matching rate slightly drops due to the limited resolution in the bounding box localization. We further observe that imperfect matching ($\tau < 1$) does not lead to significant gains in the matching rate. This is attributed to the fact that bounding boxes are well-localized between the left and right lanes with errors primarily occurring when vehicles are at the boundaries of regions in the same lane (either between regions in the same lane or at the cFoV boundaries). From Fig. 10(a), we set $\tau = 1$ when $|R| = 1$ and $\tau = 0.5$ for $|R| = 2, 4$.

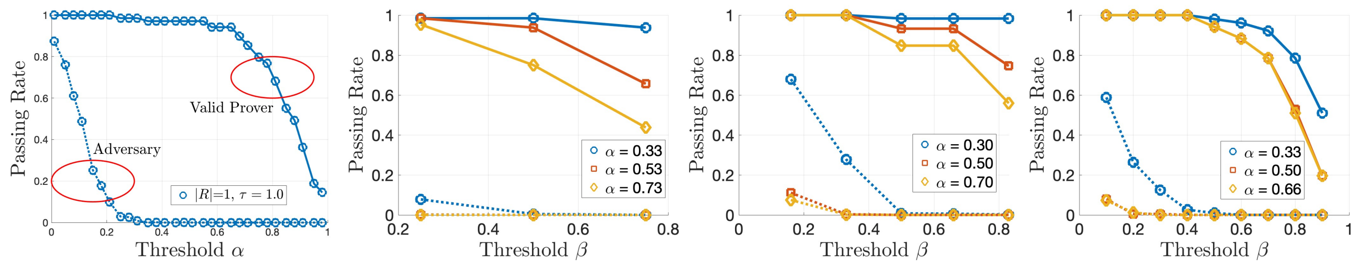
In Fig. 10(b), we show the scene matching rate as a function of the distance d_{ref} between the $\mathcal{V}$ and $\mathcal{P}$. We observe that the matching rate increases with distance because the cFoV becomes larger and bounding boxes are localized more accurately. Performance plateaus at 40m (and even slightly decreases at 45m) due to loss of vehicle detection accuracy when the prover is too far away from the verifier. As expected, tests with four regions provide the worst matching rate, however, they also yield the highest randomness because the adversary must guess the existence or absence of vehicles in each of the regions. For typical following distances and $|R| = 1, 2$ the matching rate remains above 0.75 which is sufficient to differentiate between a legitimate prover and an adversary.

Selecting α, β, K, and k. To select the remaining test parameters, we evaluated the overall *Cyclops* passing rate under benign and adversarial scenarios. The adversary was implemented using the guessing strategy described in Section V. The adversary guessed scenes according to a walk on the Markov chain generated from the traffic distribution which was assumed to be known. The verification test was applied to each of those sequences and results were averaged over 100 sequence realizations. Figure 10(c) shows the passing rate as a function of threshold α for tests containing 30 scenes when all scenes are grouped under a single test (therefore $\beta = 1$). It is noted that for $\alpha > 0.5$ the adversary's passing probability becomes close to zero for all partitions of the cFoV into regions. At the same time, a legitimate prover passes with a probability above 80%. One and two regions yield the highest passing rate, but four regions yield the strongest security (lowest passing rate for the adversary). To improve the test performance, we re-arranged the 30-scene tests to 5 groups of 6 scenes each and fixed $\alpha = 0.5$. Figure 10(d) shows the passing rates of the adversary and a valid prover. By selecting $\beta = 0.4$, a valid prover passes the test with a



(a) (b) (c) $K = 1$ and $k = 30$ (d) $K = 5$, $k = 6$, and $\alpha = 0.5$

Fig. 10: Scene matching and verification test passing rates under the uniform traffic model based on CARLA simulations.



(a) $K = 1$ and $k = 60$ (b) $K = 4$ and $k = 15$ (c) $K = 6$ and $k = 10$ (d) $K = 10$ and $k = 6$

Fig. 11: Passing rate for a valid prover and a remote adversary for the real traffic model when $|R|$ = 1 and τ = 1.

probability around 90% for $|R| = 1, 2$ and 80% for $|R| = 4$, whereas the adversary's probability is driven to zero.

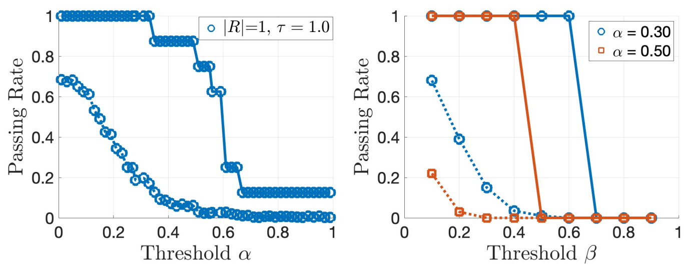
Urban Traffic Model. To further validate the parameter selection, we repeated our CARLA simulations when traffic was generated according to the urban dataset collected during our on-road experiments. Figure 11 shows the passing rate for a valid prover and a remote adversary for different K and k. We have set $|R| = 1$ and $\tau = 1$ because only one adjacent lane was consistently available in our on-road experiments. We observe that for each K, k combination, we can find a value of β (α in the case of $K = 1$) that drives the passing rate of the adversary to zero while maintaining a high passing rate for a valid prover. For instance, when all scenes belong to a single test (Fig. 11(a)), we can set $\alpha = 0.33$ and for $K = 4$ (Fig. 11(b)), we can set $\alpha = 0.33$ and $\beta = 0.5$. The same values can be used for $K = 6$ and $K = 10$, as is evident from Figs. 11(c) and 11(d). It is interesting to note that when setting $\alpha = 0.33$ and $\beta = 0.5$ the test requires less than 50% of the scene pairs between a valid prover and a verifier to match. Although 50% matching is low from a computer vision standpoint, it is sufficient to create a security advantage for the valid prover compared to a remote adversary.

Several threshold choices emerge from the simulations. One viable set is to use $(\tau, \alpha, \beta) = (1, 0.33, 0.5)$ whereas another is to use $(\tau, \alpha, \beta) = (1, 0.5, 0.4)$ ($\alpha = 0.3$ when $k = 10$). Moreover, utilizing multiple short tests as opposed to one long test provides stronger security guarantees due to post-collection data decorrelation. We examine the use of these value sets to the scenes collected from real-world experiments.

C. Evaluation in Real-World Experiments

Experimental Setup. For our real-world on-road experiments, we used an Infinity G35 representing $\mathcal{V}$ and a Honda Civic acting as $\mathcal{P}$. Two iPhone 12 Pros were mounted on the rear of $\mathcal{V}$ and the rear-view mirror of $\mathcal{P}$ facing forward, respectively. Both devices ran the Sensor Logger application, which was configured to capture telemetry data, including phone orientation. The prover led the verifier in varying traffic conditions in an urban setting (location is not disclosed to maintain author anonymity). Images were geotagged and timestamped to allow for scene synchronization and the measurement of the relative distance. We set $|R| = 1$ and $\tau = 1$ because the number of adjacent lanes varied with the roads traversed. Some representative scenes captured from the testbed are shown in Figs. 4 and 6. The two vehicles captured scenes for a total of 9 minutes at a 30 fps rate. The scenes were later downsampled to 10 fps to match the sampling rate used in the simulations. We further applied a scene filtration method that excluded scenes where the verifier and prover were stopped at a traffic light, recorded empty scenes, or the verifier's YOLO classifier lost track of the prover. Tracking of the prover was employed since the verification test involves multiple scenes over a period where the prover could sometimes be undetectable due to blockage from other vehicles. We applied a simple intersection-over-union measure for the prover's bounding box between two successive frames (1/30s apart) to keep track of the prover. If the IoU exceeded 92%, the frame was accepted, otherwise, the frame was rejected.

Evaluation. Figure 12(a) shows the passing rate for a valid prover and a remote adversary when 100 scenes are used in a single test. The pairs of scenes used in the test were increased from 60 scenes in simulations to 100 frames in the urban experiments to achieve better performance. However, we observe that for $K = 1$ there is no good choice for α that will simultaneously achieve a low passing rate for the adversary



(a) $K = 1$ and $k = 100$ (b) $K = 10$ and $k = 10$

Fig. 12: Passing rate for a valid prover and a remote adversary for the real-world on-road experiments.

and a high passing rate for any legitimate user. We further explored the setting where $K = 10$ and $k = 10$ to increase the number of tests that the adversary must pass. Figure 12(b) shows the passing rate as a function of β for different (K, k) pairs. We observe that both sets of thresholds determined from the CARLA simulations ($(\tau, \alpha, \beta) = (1, 0.3, 0.5)$ and $(\tau, \alpha, \beta) = (1, 0.5, 0.4)$) provide adequate security while maintaining the protocol correctness.

We highlight that the real-world evaluations did not achieve the same performance as the CARLA simulations due to challenges in the precise tracking of the camera coordinates in the real world (GPS resolution and camera movement due to road conditions), which impacted the cFoV and the region calculations. Despite these challenges, the desired level of security was achieved at the expense of longer delays due to the longer scene collection times.

VII. Conclusion and Future Work

We developed *Cyclops*, a CPCR protocol that binds a vehicle's digital with its physical identity, represented by its physical trajectory. In *Cyclops*, a prover and a verifier synchronously capture scenes as they travel over the claimed trajectories using low-cost monocular cameras. Identity binding is achieved by matching targets in the cFoV between the prover and the verifier. Security against a remote adversary who aims at spoofing phantom vehicles or hijacking the identity of nearby vehicles is drawn from the spatiotemporal randomness of traffic.

Future work. *Cyclops* can be improved in several ways. The use of monocular cameras presented computer vision challenges in the accurate matching of targets due to perspective distortions and poor depth estimation in angles away from the forward direction. The proposed fuzzy matching algorithm maintained security, however, it led to longer verification times. More accurate target localization using other modalities (LiDAR, Radar, or multi-camera systems) can dramatically reduce the number of scenes required for verification (even to one scene). Moreover, *Cyclops* only relies on vehicle targets (bounding boxes) without analyzing other image features such as buildings, pedestrians, license plates, trees, etc. Those features can drastically increase the scene entropy, leading to stronger security. Finally, a deeper security analysis can be performed, by exploring the Markovian properties (or lack of) of various traffic models, their collision probabilities, and using entire target trajectories as states.

VIII. Acknowledgements

This work was conducted in part during the 2022 CAT Vehicle Summer REU program at the University of Arizona and was supported in part by the US National Science Foundation (NSF) under NSF grant IIS-1950359. The authors would like to thank graduate mentors Ziqi Xu and Jingcheng Li for their mentorship.

References

[1] Q. Yang, S. Fu, H. Wang, and H. Fang, "Machine-learning-enabled cooperative perception for connected autonomous vehicles: Challenges and opportunities," *IEEE Network*, vol. 35, no. 3, pp. 96–101, 2021.
[2] Y. Wu, K. H. Low, and C. Lv, "Cooperative path planning for heterogeneous unmanned vehicles in a search-and-track mission aiming at an underwater target," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6782–6787, 2020.
[3] G. Thandavarayan, M. Sepulcre, and J. Gozalvez, "Generation of cooperative perception messages for connected and automated vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 336–16 341, 2020.
[4] I. Mahdinia, R. Arvin, A. J. Khattak, and A. Ghiasi, "Safety, energy, and emissions impacts of adaptive cruise control and cooperative adaptive cruise control," *Transportation Research Record*, vol. 2674, no. 6, pp. 253–267, 2020.
[5] R. Hajiloo, M. Abroshan, A. Khajepour, A. Kasaiezadeh, and S.-K. Chen, "Integrated steering and differential braking for emergency collision avoidance in autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 3167–3178, 2020.
[6] G. Wood, "Truck platooning expected to make inroads in 2020," https://www.oemoffhighway.com/electronics/smart-systems/automated-systems/article/21114230/truck-platooning-expected-to-make-inroads-in-2020.
[7] D. Zhang, Y.-P. Shen, S.-Q. Zhou, X.-W. Dong, and L. Yu, "Distributed secure platoon control of connected vehicles subject to DoS attack: Theory and application," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.
[8] S. J. Taylor, F. Ahmad, H. N. Nguyen, S. A. Shaikh, D. Evans, and D. Price, "Vehicular platoon communication: Cybersecurity threats and open challenges," in *Proc. of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, 2021, pp. 19–26.
[9] *IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Certificate Management Interfaces for End Entities*, IEEE Std. IEEE 1609.2.1, 2020.
[10] *3rd Generation Partnership Project;Technical Specification Group Services and System Aspects;Security aspect for LTE support of Vehicle-to-Everything (V2X) services Rel-16, V16.0.0*, 3GPP Std. TS 33.185, Jul. 2020.
[11] G. Avoine, M. A. Bingöl, I. Boureanu, S. čapkun, G. Hancke, S. Kardaş, C. H. Kim, C. Lauradoux, B. Martin, J. Munilla, A. Peinado, K. B. Rasmussen, D. Singelée, A. Tchamkerten, R. Trujillo-Rasua, and S. Vaudenay, "Security of distance-bounding: A survey," *ACM Comput. Surv.*, vol. 51, no. 5, 2018.
[12] N. Ghose, K. Gupta, L. Lazos, M. Li, Z. Xu, and J. Li, "ZITA: zero-interaction two-factor authentication using contact traces and in-band proximity verification," *IEEE Transactions on Mobile Computing*, 2023.
[13] Y. Ren, P. Wen, H. Liu, Z. Zheng, Y. Chen, P. Huang, and H. Li, "Proximity-echo: Secure two factor authentication using active sound sensing," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, 2021, pp. 1–10.
[14] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Mandayam, "Proximate: proximity-based secure pairing using ambient wireless signals," in *Proc. of MobiSys*, 2011, pp. 211–224.
[15] M. Miettinen, N. Asokan, T. D. Nguyen, A.-R. Sadeghi, and M. Sobhani, "Context-based zero-interaction pairing and key evolution for advanced personal devices," in *Proc. of CCS*, 2014, pp. 880–891.
[16] X. Li, Q. Zeng, L. Luo, and T. Luo, "T2pair: Secure and usable pairing for heterogeneous iot devices," in *Proc. of the CCS Conference*, 2020, pp. 309–323.

[17] N. O. Tippenhauer, H. Luecken, M. Kuhn, and S. Capkun, "UWB rapid-bit-exchange system for distance bounding," in *Proc. of the WiSec Conference*, 2015, pp. 1–12.

[18] S. So, J. Petit, and D. Starobinski, "Physical layer plausibility checks for misbehavior detection in V2X networks," in *Proc. of the WiSec Conference*, 2019, pp. 84–93.

[19] V.-L. Nguyen, P.-C. Lin, and R.-H. Hwang, "Enhancing misbehavior detection in 5G vehicle-to-vehicle communications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9417–9430, 2020.

[20] J. Kamel, M. R. Ansari, J. Petit, A. Kaiser, I. B. Jemaa, and P. Urien, "Simulation framework for misbehavior detection in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6631–6643, 2020.

[21] M. Sun, Y. Man, M. Li, and R. Gerdes, "SVM: secure vehicle motion verification with a single wireless receiver," in *Proc. of WiSec*, 2020, pp. 65–76.

[22] Z. Xu, J. Li, Y. Pan, L. Lazos, M. Li, and N. Ghose, "PoF: proof-of-following for vehicle platoons," *Network and Distributed Systems Security (NDSS) Symposium*, 2022.

[23] J. Han, M. Harishankar, X. Wang, A. J. Chung, and P. Tague, "Convoy: Physical context verification for vehicle platoon admission," in *Proc. of HotMobile*, 2017, pp. 73–78.

[24] C. Dickey, C. Smith, Q. Johnson, J. Li, Z. Xu, L. Lazos, and M. Li, "Wiggle: Physical challenge-response verification of vehicle platooning," in *Proc. of the International Conference on Computing, Networking and Communications (ICNC)*, 2023, pp. 54–60.

[25] C. Vaas, M. Juuti, N. Asokan, and I. Martinovic, "Get in line: Ongoing co-presence verification of a vehicle formation based on driving trajectories," in *Proc. of the IEEE EuroS&P*, 2018, pp. 199–213.

[26] M. Juuti, C. Vaas, I. Sluganovic, H. Liljestrand, N. Asokan, and I. Martinovic, "Stash: Securing transparent authentication schemes using prover-side proximity verification," in *Proc. of the IEEE SECON Conference*, 2017, pp. 1–9.

[27] (2023) The cyclops protocol. [Online]. Available: https://anonymous.4open.science/r/Cyclops-BC0F/readme.md

[28] Ultralytics, "YOLOv5: A state-of-the-art real-time object detection system," https://docs.ultralytics.com, 2021.

[29] A. Anwar, "What are intrinsic and extrinsic camera parameters in computer vision?" =https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec, 2022.

[30] M. Boneh, "Vehicle detection," https://github.com/MaryamBoneh/Vehicle-Detection, 2022.

[31] Y. He, L. Ma, J. Cui, Z. Yan, G. Xing, S. Wang, Q. Hu, and C. Pan, "AutoMatch: leveraging traffic camera to improve perception and localization of autonomous vehicles," in *Proc. of the 20th ACM Conference on Embedded Networked Sensor Systems*, 2022, pp. 16–30.

[32] N. Gadhiwala, "Intersections and collisions of simple random walks in d," *preprint, Univ. Chicago Math*, 2020.

[33] M. T. Barlow, Y. Peres, and P. Sousi, "Collisions of random walks," in *Annales de l'IHP Probabilités et statistiques*, vol. 48, no. 4, 2012, pp. 922–946.

[34] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.