Kütahya Dumlupınar University
Institute of Graduate Studies

Journal of Scientific
Reports-A
E-ISSN: 2687-6167
Number 49, June 2022

RESEARCH ARTICLE

LOCATION PROOFING USING VIDEO SIMILARITY

Cavide Balkı GEMİRTER[1,*], Tacha SERIF[2]

[1*]Yeditepe University, Division of Computer Engineering, cavidebalki.gemirter@std.yeditepe.edu.tr, ORCID: 0000-0003-3534-3129
[2]Yeditepe University, Division of Computer Engineering, tserif@cse.yeditepe.edu.tr, ORCID: 0000-0003-1819-4926

Receive Date:08.05.2022 *Accepted Date: 16.06.2022*

ABSTRACT

With the increasing availability of mobile devices and technological improvements, location-based services have become a vital part of our everyday lives. However, a layer of checks and verifications might be required to identify the user's declared location authenticity, since it is possible to bypass GPS or any other indoor location detection solutions. Existing location proofing solutions mainly propose techniques requiring some sort of an infrastructure, such as access points and/or beacons, or a co-located prover, witnesses and verifier formation to prove the presence of the user. This paper proposes a location proofing solution using the video similarity technique, which is based on the comparison of visual similarities in video pairs to determine the surrounding environment without any infrastructural overhead. The indoor location test results of our prototype indicate that it can achieve a verification accuracy of 97.05% on average in 9.67 seconds.

Keywords: *Location Proofing, Indoor Location Proofing, Video Similarity, Location-Based Services*

1. INTRODUCTION

In recent years, the growing smartphone market and innovation in mobile technologies have increased the importance of Location-Based Services (LBS) used by applications in many different fields such as finance, public services, and entertainment. For finance applications such as mobile payment systems or ATM withdrawals, verifying the user's physical location during the transaction request is of utmost importance - e.g. confirming that the customer is in front of the ATM machine. On the other hand, in assistive applications such as Foursquare, Google Maps, and Instagram, preventing fake check-ins is essential to eliminate fraudulent content introduction or inclusion.

Location proofs are plain items that enable the development of mobile applications requiring proof of the user location. A location proof is a piece of data that warrants a prover to a geographical place [1,2]. Bearing in mind that most people nowadays carry their cell phones with them all day long. To determine the behaviors and habits of these individuals, using their geolocation over Global Navigation Satellite Systems (GNSS) is a popular way of keeping track of mobile devices [3, 4]. The captured geolocation data needs an external validation mechanism because creating mock locations in these systems is a trivial task [5].

JSR A
Journal of Scientific Reports

Conventionally, in location proofing solutions [6-12], the prover generally requests a Location Proof (LP) of her/his current position, where an LP is a certified meta-data containing the geographical location [6]. On the other hand, the LP verification mechanism that facilitates this service can either be centralized or decentralized. Centralized architectures [6,9,13] estimate the mobile user's location proximity by relying on wireless components such as a Wi-Fi Access Point (AP) or Bluetooth signal emitter devices [14,15]. However, on the contrary, the decentralized architectures [7,8,10-12,16-19] do not utilize central points, and every mobile device in the network acts as a witness to confirm the locations of other mobile entities. Though, in decentralized architecture, Prover-to-Prover (P-P) and Prover-to-Witness (P-W) collusions are the non-trivial, hard-to-solve security issues, in addition to the privacy concerns, which are created as a result of stored user location data. In a P-P collusion, a prover requests proof for a location where the attacker is not actually present. In a P-W collusion, the dishonest witness generates proof for a location where the witness, the prover, or both are not present.

Maia and Pardal [22] designed a location proof system, called CROSS, relying on Wi-Fi access points. The prototype is a location proofing system implemented as part of a reward-based tourism application scenario, keeping track of users traversing the provided route. The authors propose three different location proof strategies, namely scavenging, Time-based One-Time Password and kiosk. The scavenging strategy is operated in urban areas, employing scan results of Basic Service Set Identifiers (BSSID) as evidence for the Location Proof (LP). BSSID is the MAC address of the Access Points (AP). Time-based One-Time Password (TOTP) strategy is built on customized Wi-Fi APs, ensuring more security by dynamically changing broadcast SSID. The kiosk strategy validates the presence of the user physically. Because the system relies on Wi-Fi for location proximity, the capabilities of the Wi-Fi devices limit the accuracy. Another constraint is that the visitors should be present for at least five minutes in the proper location to guarantee the correctness of the verification results.

From a different perspective, Santos et al. [23] implemented a vehicle tracking system called Secure Transport lOcation Proofs (STOP), which is aimed to identify vehicles averting from their predefined track. In this solution, every truck driver is expected to utilize his/her mobile phone to log its movement coordinates using GPS data. Then, these collected logs can be used by inspectors to audit the truck's movements so far. The inspectors can use their own mobile devices to contact the driver's data via short-range Bluetooth and download all the logs. The evaluation results of the prototype show that the GPS quality is reduced in urban and densely populated areas due to the limitation of GPS, which requires line-of-sight for best performance. Hence, the worst physical locations from this point of view are the tunnels, which are blocks of concrete where phones totally disconnect from GPS satellites. The average error distance from the exact rotation is calculated between 5 and 9 meters, and the worst case has an error of up to 18.97 meters.

From a different perspective, Hasan and Burns [13] investigate the secure location provenance problem and propose a witness-based solution that generates collusion-resistant location proofs. In this proposed prototype, the user, who needs location verification, sends a request to a publicly accessible location authority, attaching the unique global identifier of the desired place. The location authority responds with a signed LP after making a simple presence check for the requester in the given place. Next, the prover searches for a closer witness through Bluetooth to endorse his LP by performing distance bounding. To guard against location history frauds, the authors also present two schemes set on hash chains and Bloom filters. Evaluation results indicate that 46 proof/second can be achieved with the Bloom filter scheme and 60 proofs/second with the hash chains. Although the

proposed solution eliminates dependency on trusted third parties, standing a centralized architecture is the drawback of the system.

Following suit, Zhu and Cao [8] proposed a neighbor-based solution that allows cooperation between co-located Bluetooth devices to generate LPs for each other. In the prototype, named Privacy-Preserving LocAtion proof Updating System (APPLAUS), the prover broadcasts the request to the neighbor devices over Bluetooth and if there is no response, the prover generates a dummy LP for submitting the updated location to a centralized verification server. If a witness agrees to the proof request of the prover, he generates a response to send back to the prover. In order to increase privacy and protect the content from untrusted nodes, devices periodically change their pseudonyms. The evaluations of the system show that the client code uses less than 2.5% of the available memory, with a CPU utilization between 3% and 5% when communicating with other devices and less than 2.5% power consumption. The authors evaluate the solution's performance for different update and contact intervals ratios. It has been revealed that if the location proof update interval is more than 1.5 of the contact interval, the performance reaches a satisfactory level - above 93%. The weakness of the solution is that it only protects against simple collusion attacks, which do not gather bogus LPs from honest witnesses. The short communication range of Bluetooth constrains the system's efficiency, and periodically generated dummy proofs to preserve privacy cause a communication overhead.

Nosouhi et al. [16] propose a decentralized, blockchain-based, secure, and privacy-aware LP generation and verification approach. The main actors are the prover, the witnesses, the bridge, and the verifier. When a witness endorses the location of the prover, he adds the selected target bridge ID to the LP and sets a timer before sending the message back to the prover after signing the LP request. The timer prevents the Prover–Prover collusions, which have a remote dishonest prover. The witness that receives the response in the given time creates the transaction and forwards it to a nearby bridge. When the bridge receives the message, it compares his identity with the bridge ID written in the message. If the response does not come back from the prover within the appropriate time, the witness sends an N-Ack message indicating a Prover–Prover collusion. In the last step, the bridge forwards the message to the verifier over the internet for doing the post signature controls of the earlier steps. In a successful transaction verification, there must be at least a set of transactions not marked as non-N-Ack. There is a reward in the system, which is the amount paid by the prover to the LP approvers in the final step, with a minimum threshold to prevent Prover–Witness collusions. The authors count the percentage of LP requests that successfully pass the P–P collusion detection for different hash key sizes and timeout system parameters in the evaluation tests. For the key sizes 1024, 2048, and 3072, the timeout parameters of 400, 500, and 700 ms are sufficient for successfully detecting most of the collusions. Nonetheless, the proposed solution is weak to the multiple-party collusion attacks.

The same authors proposed another witness-oriented, decentralized approach where witnesses generate the LPs for other mobile users in ad hoc environments [17]. In addition to dealing with Prover–Prover and Prover–Witness collusion attacks, the authors presented the P-TREAD protocol to increase the system's privacy in the implemented prototype, Privacy-Aware and Secure Proof Of pRoximiTy (PASPORT). Although PASPORT moves the control of witness selection from the prover to the verifier, it is still open to three-way/multi-party collusion attacks.

Zafar et al. [19] focus on three-way collusions, which are not resisted by previous studies. In these collusions, only a single party makes the selection of participants, causing collusion attacks probable. To decrease the chance of location manipulation, the researchers implemented a decentralized and

secure consensus protocol called MobChain. The system separates the choosing control of participants with the LP generation steps. The witness and location authority selection is made in a private blockchain where the nodes are peer-to-peer connected. Because the proposed approach offers substantial secure contracts, falsifying the location becomes impractical. In the evaluation tests, MobChain enhances the protection of LPS in a computationally efficient and highly available way compared with state-of-the-art techniques.

Diversely different from prior studies focusing on infrastructural system changes, some researchers draw attention to the advantage of the media for location proofing. Bucher et al. [21] presented a cheap to implement system built on visual features using image recognition with no infrastructural overhead. There are three leading roles in the architecture as follows: (1) LBS Provider like a web application, (2) location owner who owns the place like a restaurant or cafe, and (3) the user client requesting the LP. When a user requests an LP, the LBS provider assigns a task to the user as taking a picture of the given location, similar to the already present image data set saved in the central server. After collecting a sufficient number of pictures for a particular place, the system quickly detects the real and bogus LP requests. The given challenge task may be one of these options: taking a new photo contributing to the image tree, an overlapping copy of an image randomly selected from the tree, or a partially overlapping image like completing half of an existing image. The system results in better performance, especially on crowd-sourced destinations where a service provider confirms the collected data's authenticity. In the evaluation tests, although the prototype cannot prove all of the users from a cryptographic security perspective, the system catches the majority of the attacks in most of the scenarios. The limitations of this approach are that at least an honest user must be contributed to the given places. Completing the given challenge will be feasible for indoor locations such as restaurants, but it will not be easy for large-scale outdoor areas like parks to find the correct part of the place to take the picture defined in the assigned task.

In the video subsection of the media domain, video filtering, video suggestion, and copyright protection are the challenging research problems. Kordopatis-Zilos et al. [24] address the problem of similarity estimation between video pairs. Previous video retrieval approaches embed the entire frame or video into a vector embedding. The implemented prototype, called Fine-grained Spatio-Temporal Video Similarity Learning (ViSiL), considers fine-grained SpatioTemporal relations to consist of intra-frame and inter-frame connections, commonly lost in previous studies. The study shares evaluation comparisons against near-duplicate video retrieval, fine-grained incident video retrieval, event video retrieval, and action video retrieval subtasks of the computer vision domain.

Extracting information from video and visual content has been an active research area since the 2000s [20]. Accordingly, compared to infrastructural system-focused approaches, employing media for location proofing could be used as another totally different solution avenue. In this approach, location proofing is achieved by comparing the visual features of two images or video frames, to check whether the two locations are the same place or not [21]. Hence, this paper proposes location proofing using the video similarity technique, which addresses the challenges mentioned above. Furthermore, it describes a novel location proofing solution based on a comparison of the visual similarities in video pairs to determine the surrounding environment without any infrastructural prerequisites. To the best of our knowledge, there is no work that has utilized video similarity evaluation techniques for location proofing purposes.

JSR A
Journal of Scientific Reports

2. MATERIAL AND METHODS

Location proofing using video similarity consists of two major parts. Employing the live video streaming ability of web or mobile clients with cameras in the source device, the user records a short panoramic video clip of the indoor environment as proof of existing in the selected location. The second part checks the validity of the recorded video, comparing it with the already collected and approved video files for the proper location. This section firstly provides information about the WebRTC [25] protocol used on the clients for live streaming, followed by the ViSiL [24] framework, which calculates the video similarity scores of multiple video files.

2.1. Web Real-Time Communication (WebRTC)

With the wide use of the internet and the increased popularity of smartphones, video streaming has become a de facto segment of communication applications such as social media, chat platforms, and video conferencing [26]. Although video communication seems straightforward, like data transfer, the most challenging characteristic of the video is that it requires a considerable bandwidth [27].

WebRTC is a state-of-the-art open technology, published by the World Wide Web Consortium (W3C) [28], a standard for end-to-end encrypted real-time communication between Internet browsers for media and data transfer. The main logic of WebRTC is that the client browsers register a common communication service provider website, as shown in Figure 1.

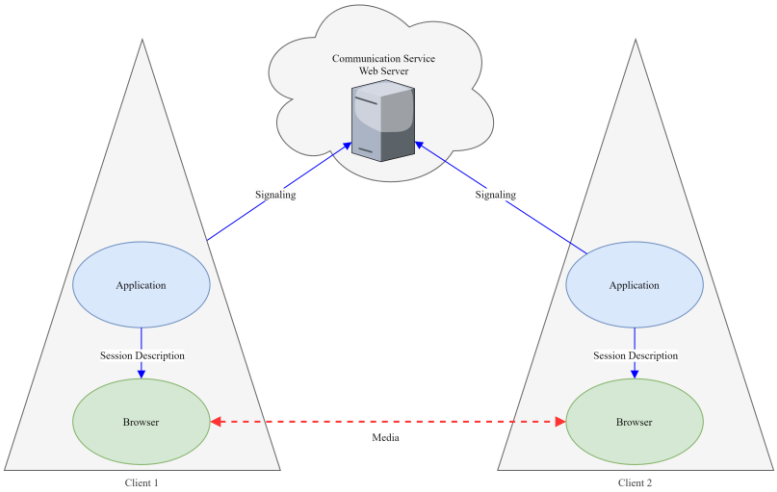

Figure 1. The main architecture of the JSEP protocol.

After the website publishes the browser IDs of the connected clients to the requester client, the client browser can establish a peer-to-peer communication with the target browser directly without knowing the exact IP addresses. To overcome the IP address and port requirement of the P2P networks, WebRTC establishes signaling channels between the website server and the clients using Web Socket technology and JavaScript Session Establishment Protocol (JSEP) [29] to create the offer and answer messages. After the proposal of WebRTC, browser vendors (Google Chrome, Mozilla Firefox, Safari, and Internet Explorer) implemented the WebRTC API for browser-to-browser communication

management and introduced a set of more easy-to-use JavaScript functions for application developers. A specific method named "getUserMedia" is used to access both the client devices' audio and video input and outputs with the user's attention for taking approval of the usage. WebRTC has three main functions. RTCPeerConnection is the service provider API for peer-to-peer communication, RTCDataChannel establishes a full-duplex data communication between the clients, and MediaStream denotes audio or video data streams. Only the MediaStream function of the WebRTC protocol is enough for capturing video frames from the client device camera, disabling the audio sources.

2.2. ViSiL: Fine-grained Spatio-temporal Video Similarity Learning

As a result of the increasing popularity of video-sharing services such as YouTube, publicly visible video content on the internet has been boosted to remarkable scales. The video recommendation and copyright reservation systems have become crucial parts of these applications.

ViSiL is a video similarity learning network for handling the similarity calculation and estimation of multiple videos. Traditional approaches group the features of all the video frames into a single video-level vector descriptor, ignoring the spatial and temporal layout of the visual similarity.

On the other side, ViSiL deals with both the spatial (intra-frame) and temporal (inter-frame) structure of the visual content when calculating the similarity to avoid ignoring the regional details. The main architecture of the video similarity calculation process is illustrated in Figure 2. First, the system extracts the feature vectors (output of Figure 2.a) of each frame of the provided video pair by utilizing a pretrained Convolutional Neural Network (CNN). After several normalization and whitening steps, the features are weighted with an attention mechanism. A similarity matrix (output of Figure 2.b) of the video pairs is built with the Tensor Dot (TD) product which computes the pairwise frame-to-frame similarity, followed by a Chamfer Similarity (CS) to calculate the region-to-region similarity without feature aggregation. TD gives the sum of tensor pairs for the given axes. CS is the similarity counter of Chamfer Distance by calculating the average similarity of a set of items for the given property. The similarity matrix (output of Figure 2.c) is the input of another four-layered CNN that has a CS final layer to summarize the video-to-video similarity score. The definitive score has a value between [-1, +1], where negative values indicate irrelevant and positive values denote relevant video pairs.

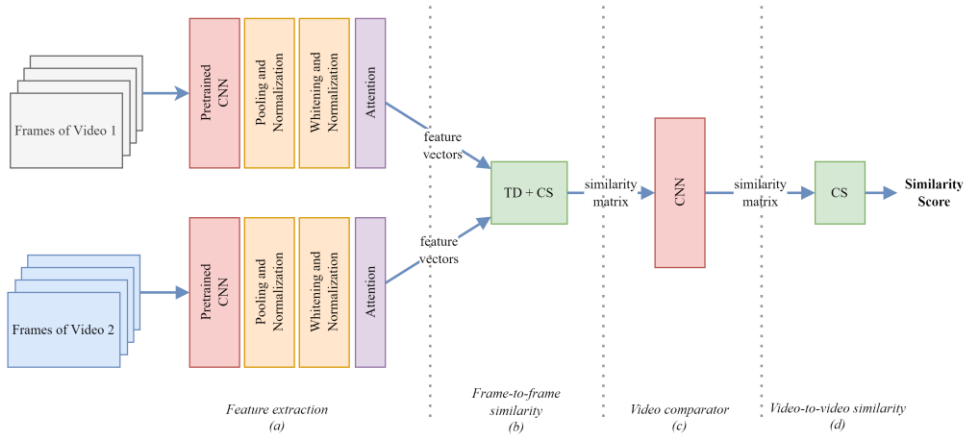

Figure 2. The main architecture of the ViSiL framework.

2.3. System Desıgn

This section discusses and examines the requirements for achieving the necessary functionality of the proposed system.

2.3.1. System analysis

The idea behind the approach is based on the similarity of video pairs recorded in several locations. The mechanism needs an application to serve the user in a mobile environment, a native mobile application, or a website running on mobile browsers like Chrome or Safari.

The video verification task is based on memory-hungry computer vision algorithms. Because the algorithm will compare the recorded video with the other video pairs, the operating environment needs to access the earlier recorded videos of the given location. Hence, the video verification operation could not be completed on the client's mobile device; a central server must handle the task, which has sufficient disk size, CPU, and memory.

An external authentication mechanism is needed to trust and identify the user, requesting the location proof. The unique user ID returned by the authentication service will be used in all the operations. After a successful login, already present locations in the system are displayed to the user for his selection. The user may select one of these options or enter a new location name.

The user’s smartphone should have a video input device to record a video as evidence of the location proof. If multiple cameras are attached to the device, the user should select one of these inputs. Modern smartphones have at least two cameras, one is in the front, and one is in the rear. The mechanism should force the rear camera to record a better panoramic video of the environment.

The resolution of the camera is another constraint on the client device. Low-resolution videos will not result in apparent feature vectors. Recorded video on the client device will be sent to the centralized server for verification. Transferring high-resolution videos will take longer and be sensitive to communication errors. So, 640x480 will be the exact resolution supported by the solution. If the client camera does not support this minimum threshold resolution, the recording operation should not be started on the device.

WebRTC is the state-of-art technology used to capture the media streams on client browsers. Most modern browsers natively support WebRTC, but there are some exceptional browsers. The browser on the client device should support the WebRTC protocol to record the video. When the web application calls the getMediaStream function of the WebRTC protocol, the browser asks the user to allow accessing the selected camera. The user's attention is required to start the capturing process in this step. In the recording phase, the audio will be muted.

Correctly defining the length of the recorded video is crucial. Longer videos will be sent to the central server in a long time, and also, the computer vision tasks in the verification phase will take much more time. Five-second long video will be sufficient to identify the video's features and calculate the similarity with other videos. If the application captures ten frames each second, there will be 50 frames in total.

While recording the video, the user should slightly swivel the camera toward the left and right for best performance. When the video clip is not recorded in a panoramic view, documenting a narrow area for

5 seconds would result in an image rather than a video, not including temporal data, and would have a weak feature spectrum compared to the video. To eliminate this problem, the algorithm should check the sanity of the videos with a static content filter.

When the location is new and there are no earlier sample videos in the repository, the system may automatically accept the video for the first a specific number of samples. Rather than automatically accepting the initial videos as marking positive samples of the given location, a manual check either be done by the place owner, who is the restaurant, or the café administrator.

Instead of continuously getting new video samples of the locations from the users, some additional controls should be added to the system to increase the security level. A challenge-based approach defined in Bucher et al.'s study may be an example. For the given location, the solution may sometimes get new video samples, or sometimes the application will assign some task to the user to complete as recording a video, including a displayed image or a video slice.

2.3.2. Solution design

The proposed system has two primary and one optional stakeholder; (1) the user who requests a location proof, (2) the location-based service provider running the algorithms in the server, and (3) the optional location owner who is the business administrator of a restaurant or café.

The sequential diagram of the proposed solution is shown in Figure 3. Initially, the user opens the web application served by the LBS provider, deployed on a centralized server. After a successful login operation done by an external authority, the web application lists the earlier locations present in the system. The user may select one of these locations or enter a new location name. Then, the user requests a challenge from the LBS provider for submitting the LP. The LBS provider has two options: request the user to record a new video or, after randomly selecting one of the earlier videos of the selected location, the LBS provider requests the user to record a video including the displayed frame. The user has two minutes to complete the task.

JSR A
Journal of Scientific Reports

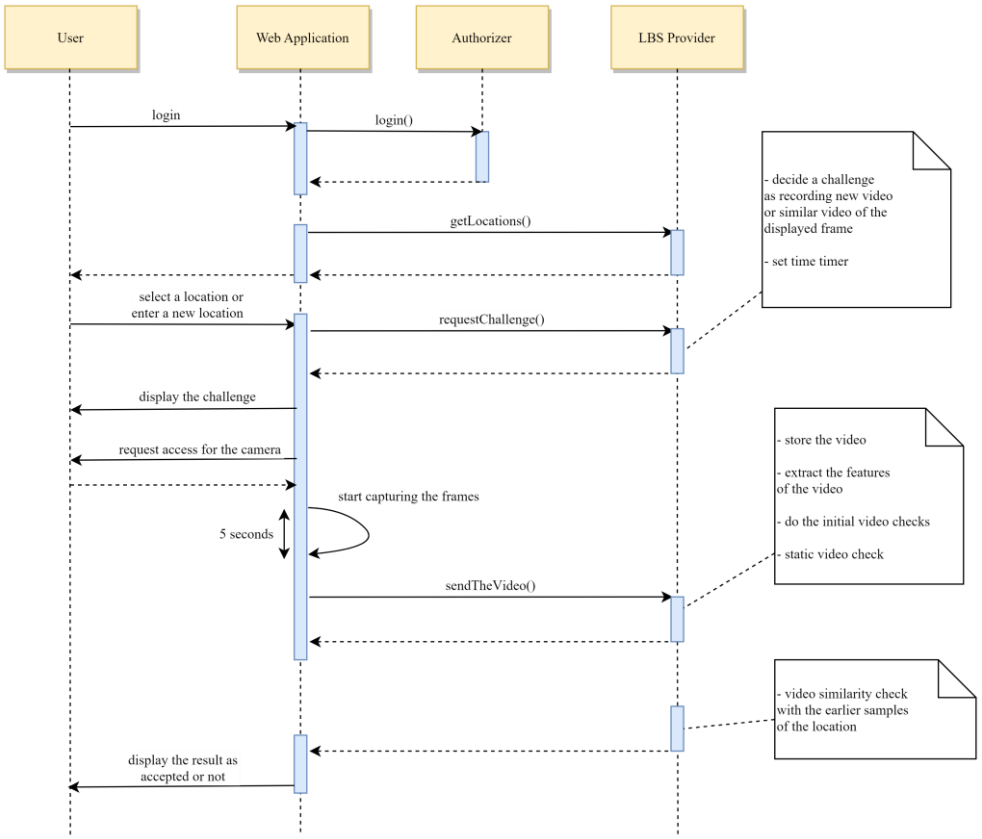

Figure 3. The sequential diagram of the proposed solution.

After defining the challenge to the user, the web application asks the user to select the video input and requests allowing access to the selecting camera. The web application captures fifty frames for five seconds while the user slightly turns the camera in the environment to have a panoramic view. The web application sends the recorded video frames to the LBS provider at the end of the recording. If the challenge is not completed in two minutes, the LBS provider automatically rejects the LP request. When the LBS provider receives the video in the given time interval, it checks the initial sanity of the video by counting the frames, extracting the feature vectors of the video followed by operating the static video filter. Then, the LBS provider saves the video for further video similarity check batch operation.

The recorded video is the evidence of the prover, which is marked as the source video of the LP request in the location. The target videos are stored attachments of already authorized LP requests of the proper location, used as similarity hints by the affinity mechanism of the system. This mechanism examines the video pairs to see if they look like each other by comparing their visual features and detects whether they are recorded in equivalent locations or not. When the similarity check is done, the result of the LP submission is displayed to the user as Accepted or Rejected.

JSR A
Journal of Scientific Reports

2.3.3. System architecture

The system architecture of the proposed solution is shown in Figure 4. The web application server publishes the web application, which serves on the client's PC or smartphone and communicates with the video input sources of the device over WebRTC technology. The web application communicates with the backend server over the internet and gets services using POST API calls. The backend server contains the LBS provider, responsible for the service operations, and the affinity system operating the video and neural network accelerated operations. Rather than a standalone database server, simple Comma Separated Values (CSV) files are used to store the status of the operations and results. Both the LBS provider and Affinity system share the common file system to store and read the video files; the LBS provider writes the video files, and the affinity system reads these files and stores the extracted feature vectors. The affinity system employs the pretrained neural networks, which extract the feature vectors of the videos and calculate the video similarities.

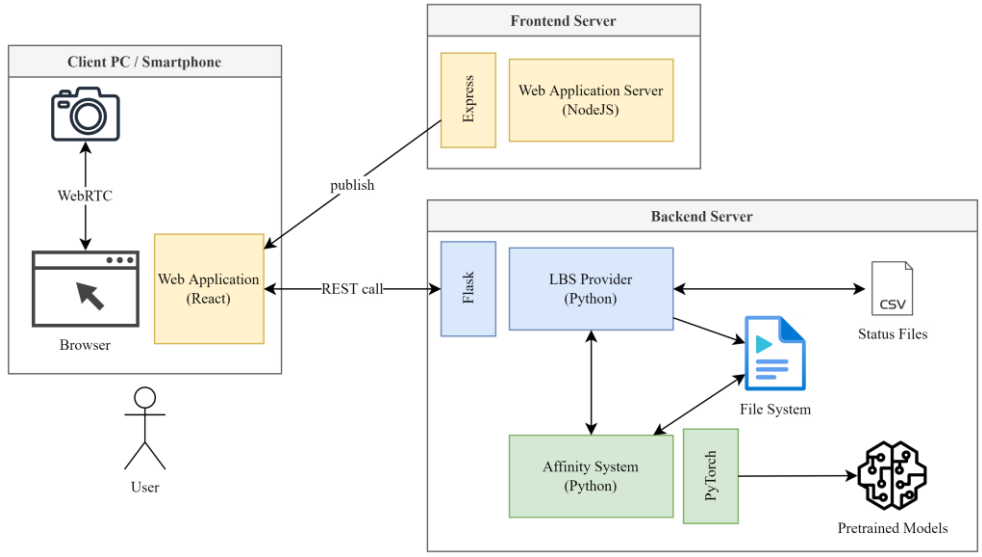

Figure 4. The system architecture of the proposed solution.

2.4. Implementation

The solution has recording and verification phases. In the recording phase, the user logins to the system, requests a location proof, and records a video to complete the assigned challenge tasks. Next, the LBS prover launches several controls over the process and checks the sanity of the recorded video. In the verification phase, the affinity mechanism runs the video similarity algorithms to approve or reject the LP request. The user can visually track earlier submitted requests and their ongoing or completed statuses from the website's history section.

2.4.1. Recording phase

The LBS provider interacts with the user over a website. The website initially redirects the user to a third-party login page; Google 2.0 Authentication algorithm is selected in the solution. After a successful login operation, the user identifier is attached to the session as the input for further

validations. The user chooses the desired location from the earlier documented locations list or enters a new place name as the location input and requests location proof from the LBS provider.

As part of the challenge assignment step, the system examines whether the location input is present in the database or it is a new location. If a new location is entered, the LBS provider requests the user to record a short panoramic video clip in the given place. If the user selects a place from the locations list, the LBS provider assigns a challenging task to the user, which is randomly selected from the following two options. In option (1), the algorithm requests the user to record a new video in the given place like the previous step. This task aims to enrich the video repository with new video visuals. In option (2), the algorithm randomly selects one of the target location's earlier registered and approved videos, as a candidate video V_C, followed by randomly selecting one of the frames F_C of this candidate video V_C. Next, the system assigns a challenge to the user, and the user has only two minutes to complete the given task. This two-option and timer-based challenge process seek to increase the security level and prevent fake user attacks by uploading bogus videos. Two minutes is long enough to complete the challenge but short to integrate the previously displayed frame into a video for faking the proposed task. When the challenge is assigned to the user, the LBS provider sets up a two minutes timer, generates a unique identifier for the generated challenge, and responds to the user.

In the video recording step, the website displays the candidate frame F_C to the user if option (2) is selected. Then, for using the live media capturing capabilities of the smartphone's browser, the web application requests the user to allow access to the environment camera of the device. If the user device has more than one camera, the system asks the user to select one of these cameras as the source. After the user gives permission, the application starts frame capturing for five seconds. The application records ten frames every second, and 50 frames are collected to build the video. Then, the application stops capturing and sends the video frames back to the server with the challenge and user identifiers.

The LBS provider first checks the status of the timer, and if it has timed out, the system automatically rejects the LP request. Secondly, the algorithm counts the frames and scans their order. After validating the challenge and user identifiers with the submitted ones in the initial LP request, the algorithm extracts the visual features of all the frames and builds a three-dimensional cube vector representing the source video. The static content filter utilizes the frame-to-frame similarity of the ViSiL approach to check if the video has a valid panoramic view. The system calculates the frame similarity of each video frame by comparing it with the rest of the frames one by one. If the average frame-to-frame similarity score exceeds the threshold system parameter, the algorithm rejects the video and marks it as holding static content.

2.4.2. Verification phase

The system sets the recorded video as the source of the similarity algorithm. If the submitted video contains a location that is new and there are no earlier sample videos of this location in the repository, the system automatically accepts the video as the first sample video - out of three. If an existing location is selected from the drop-down menu and option one (1) is picked by the system, the algorithm sets all the earlier recorded samples of the chosen location as target videos. In this way, the system compares the similarity of the latest recorded video with the rest of the earlier samples. If option (2) is picked by the system, then video V_C, which contains the randomly chosen candidate frame F_C, partially or fully is set as the target video. Then, the algorithm only compares the similarity of the recorded video with V_C, which could either be the full length of the video or a part of it. Trials

development phase showed that one-second video is long enough to be sure that the video is recorded in the same place and short enough not to force users to shoot the same video with the V_C.

After defining the source and target videos, the system performs the video similarity check, operating the methods defined in the ViSiL framework. Initially, the visual features are extracted by applying a pretrained Convolutional Neural Network (CNN) to the frames of the video pairs. Initially, for each video in the pair, the visual features are extracted from the frames by involving a pretrained Convolutional Neural Network (CNN). After several normalization steps, the attention mechanism weighs the features. Tensor Dot (TD) product is applied to measure the pairwise frame-to-frame similarity between the source and target videos, which outputs a similarity matrix by summarizing the tensor pairs on proper axes. Next, Chamfer Similarity (CS) employs this similarity matrix and computes the region-to-region similarity by counting the Chamfer Distance without feature aggregation. Chamfer Distance is the average similarity of the items set for the given property. The last step is another four-layered CNN with a CS final layer which obtains the similarity matrix and summarizes the video-to-video similarity score. The conclusive score has a value between [-1, +1], where negative numbers point to irrelevant and positive numbers distinguish relevant video pairs. If there are numerous target videos, the system independently calculates the similarity of each target video with the source video and brings the maximum value as the final decision.

After the affinity mechanism completes the similarity checks of the recorded video, the LBS provider updates the status of the LP request historical record in the system as Accepted or Rejected. For evaluating the sanity of the decision, the web application opens a survey to the user whether the given result is correct or not. These grading results are collected and analyzed in the evaluation results.

2.4.3. Development details

The system has three development components: (1) the web application serving the user, (2) the LBS provider as the backend server, and (3) the affinity system responsible for video operations and similarity checks. The web application is developed with React 4+ and deployed with Express modules on a NodeJS server. React is selected because it natively supports a vast spectrum of modern browsers. LBS provider is developed with Python 3.10 and serves as again a web server using Flask modules. Instead of using a standalone database server, simple CSV files, maintained with Pandas, are used to store the status of the system and the operations. The affinity system is built inside the LBS provider, and the computer vision components are implemented using the Python Pytorch library. The web application communicates with the LBS provider over the internet and gets services over POST API calls. The topology of the system is shown in Figure 5. Because the affinity system is a memory-hungry and computationally costly component, the web application server and LBS provider are separately deployed on different servers. Both LBS provider and affinity system is deployed on the same server to access the same file system which hosts the stored video files and feature vectors.

JSR A
Journal of Scientific Reports

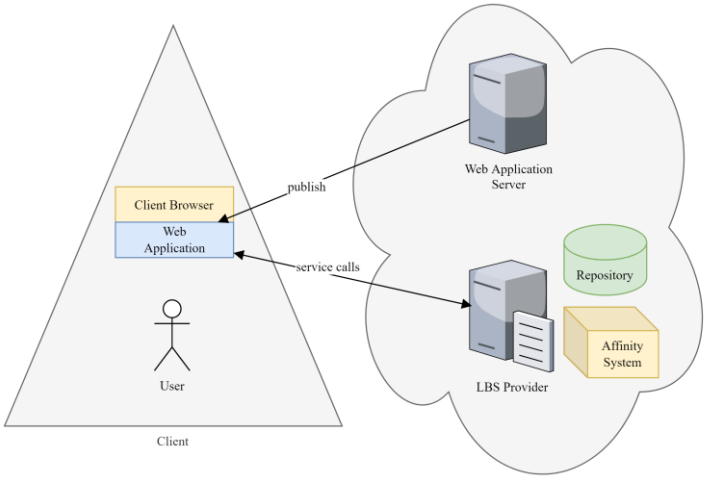

Figure 5. The topology of the proposed solution.

3. RESULTS AND DISCUSSION

3.1. Experimental Setup

The implemented system components are deployed on cloud platforms for evaluation tests. The web application server is deployed on a free dyno of the Heroku service platform, located in the United States, with a slug size of 68.2 MB. The LBS provider is deployed on an e2-medium machine typed Google Cloud compute engine, with one vCPU, 4 GiB memory, Ubuntu 18.04 operating system, located on us-central1-a zone. The clients are located in Istanbul (Turkey) and Bologna (Italy). Apple iPhone 11, iPhone X, iPhone 7 and General Mobile 9 Pro smartphones are used as the clients in the tests. For iOS-based clients, both the Safari and Chrome, and for Android typed phones, only the Chrome browser is operated as the browser.

3.2. Evaluation Results

3.2.1. CPU and memory utilization

In order to evaluate resource usage of the LBS provider, CPU and memory usages are captured from the server where the component is deployed. For an accurate evaluation, only the LBS provider is deployed as a single-process application on the cloud virtual machine to minimize the effect of other threads. The captured CPU and memory usage are shown in Figure 6. There are some peaks in the CPU usage when the affinity system processes are active, like extracting the video features and calculating the video similarity of video pairs. CPU usage significantly drops to 0% when the system is idle. The average CPU usage of the system is 2.214%, with a minimum value of 0% and a maximum value of 89.1%. Unlike CPU, memory usage is not sensitive to the affinity system threads. After the predefined models are loaded during the initialization step of the application, the memory usage is increased to 18.5%. It is increased to 26.7% on active video operation threads. The average memory usage of the system is 23.91%, with a minimum value of 18.5%.

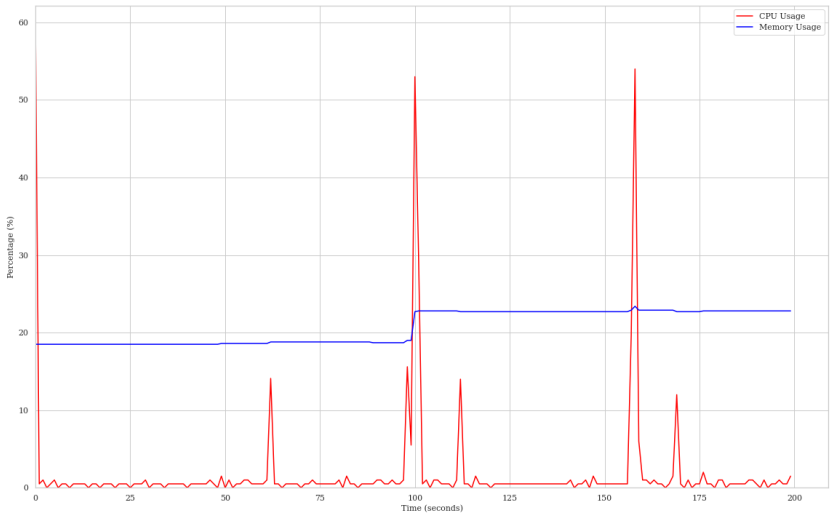

Figure 6. The CPU and Memory usage of the LBS provider.

3.2.2. Processing time of the operations

This section evaluates the processing time of the client and server operations. The most time-consuming operation in the process is uploading the videos from the client to the server. The duration is directly related to the network communication quality between the client and the cloud server location in the U.S. The upload time increases up to 20 seconds when the connection bandwidth is very low. For a 5-seconds long video having a resolution of 480x640, the average upload video time of the client is 9 seconds, with a minimum value of 5.75 and a maximum value of 19.79 seconds. The operation time likewise doubled when the video length is increased from 5 seconds to 10 seconds. There is even a direct positive proportion between the resolution of the video and the uploading time. When the resolution is increased to 768x1024, the time duration is expanded from 9 seconds to 15 seconds.

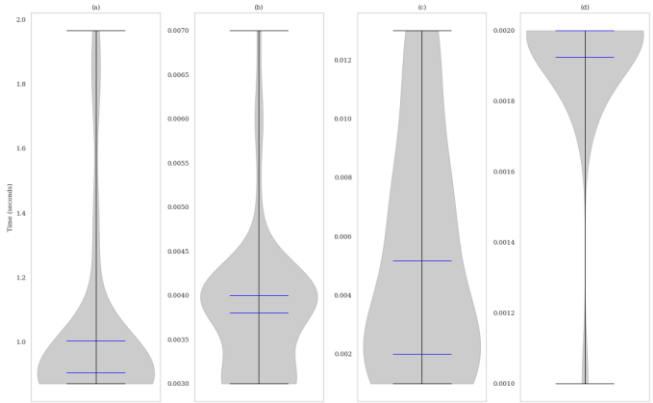

Figure 7. The processing time of operations. (a) Extract features. (b) Static video check. (c) Video similarity check. (d) Video similarity check (normalized).

There are three time-consuming primary operations done on the server-side, shown in Figure 7. The first violin shape illustrates the time taken to extract the features of the videos. In an only CPU environment, the average processing time of extract feature operation is 1.003 seconds, with a minimum value of 0.871 seconds and a maximum value of 1.966 seconds. Compared to the other affinity system jobs, this is the most time-consuming computer vision accelerated step that employs the pretrained CNN module. If the server has a GPU (Tesla V100) and the operation is done with a batch size of eight, the duration significantly drops to 0.5 seconds on average.

The second shape represents the time duration of the static video review operation. For a five-second long video, the operation takes 0.003 seconds on average, with a minimum value of 0.003 seconds and a maximum value of 0.007. The standard deviation of the time is very low compared with the other affinity system processes. Operating GPUs does not result in a significant improvement in the processing time.

The last two violins illustrate the processing time of video similarity operation. There are two types of challenges in the solution. The user may upload a new video to the system, and the affinity system compares the video with all of the other videos already recorded and approved in the given location. In this challenge, the video similarity examination operation is repeated n times, where n is the number of earlier captured videos in the location. In the second option, the client records a video, including the displayed frame, to the user. The video similarity operation is done only once in this scenario. The third violin represents the all-time taken operation for the video similarity check operation, and the last one is the normalized version of the time in which the total time is plotted in (c) is divided by the video count, indicated as n. For a five-second long video, the operation takes 0.005 seconds on average, with a minimum value of 0.001 seconds and a maximum value of 0.013. When the data is normalized, the average time is measured as 0.001 seconds, with a minimum value of 0.001 seconds and a maximum value of 0.002.

3.2.3. Accuracy results of the verification

After displaying the verification result of the location proof operation, the prototype asks the user to grade the response as correct or wrong. Depending on these human testing results, the accuracy of the affinity system is calculated. For the static video review step, the accuracy is measured as 98.2%. The algorithm only fails for the videos when the user moves the smartphone camera more slowly than the expected threshold, during the recording step.

For the first option type of challenge - a user uploading a new video for the given location - if the threshold system parameter is set to 0.8, the algorithm's accuracy is 85.0%. On the other hand, if the threshold parameter is decreased to 0.5, the accuracy increases to 99.5%. For the second option challenges, the accuracy is decreased to 71.4% when the target video is set to a one-second-long sub video, including the displayed candidate frame. The accuracy is significantly increased up to 92.85% when the target video length is set to three seconds and 99.5% when the target video is not cropped. The average accuracy of the video similarity check operation is calculated as 97.05% in summary.

The algorithm fails when the video includes many irrelevant features that mislead the system to identify the location environment, like humans or static visual parts such as empty walls in the rooms. Compared to outdoor environments, the algorithm gives better results in indoor environments. When the prototype is tested on a national park where there are many similar patterns like the lakes, trees, or the grasses, the affinity system video similarity check algorithm accuracy goes down.

4. CONCLUSIONS

This paper presents a novel centralized architecture for location proofing, which contrary to earlier studies, does not need or require any sort of hardware or infrastructure. The proposed approach is based on comparing visual similarities in video pairs to determine the surrounding environment. The affinity system built in the LBS provider has two main parts. Operating the live video streaming ability of web or mobile clients with cameras in the source device, the user records a short panoramic video clip of the indoor environment as proof of existing in the selected location. The second part checks the validity of the recorded video, comparing it with the already collected and approved video files for the proper location. Because the system does not require any additional settings on the client-side and customized network components placed in the locations, it can be easily integrated into daily life location proof required scenarios.

The implemented prototype system can perform with limited CPU needs, where the average CPU use rate is 2.21%, and the maximum use rate is 89.1%. Furthermore, the memory consumption does not go beyond the 26.7% limit, and the time needed for the server to extract the features of a video is 1.003 seconds on average. Extraction time could be even reduced by 0.5 seconds with the use of a GPU. On the other hand, on a client device, it takes on average 9 seconds for a 5-seconds long video with a resolution of 480x640 to be uploaded to the server. Overall, the prototype's similarity check accuracy is 97.05% in an indoor environment. Bearing in mind the above results, it is clear that this type of solution would fit great in a scenario where the user is aiming to prove the location of an indoor environment with no physical networking infrastructure. Considering the latest move from popular streaming services, where they try to prevent their subscribers from accessing content from multiple sites, integrating a user location proving system would be an effective way to achieve this goal. Accordingly, the solution proposed as part of this work can facilitate a control mechanism without requiring any additional software or hardware infrastructure.

Like all other location proofing systems, the proposed method has its shortcomings too. This prototype accepts the first three samples of a newly registered location without conducting any checks and accepts them as is. If a dishonest user creates a new location and uploads an initial couple of bogus videos, there are no additional control mechanisms to prevent this. Also, need to be considered the cases where the user moves the smartphone camera slower than the expected threshold while recording; this results in the static video check algorithm failing. Furthermore, the video similarity algorithm may not perform as required if the video includes notable unrelated features that mislead the system in identifying the location environment - e.g. too many people in a closed environment or blank patterns. As a result of this, compared to outdoor environments, which may be crowded with humans, objects, and blank patterns, the algorithm performs better in indoors. The most noteworthy drawbacks of the system are not having a satisfactory success rate in outdoor environments and automatically accepting the first three samples in a new location without any control. In order to increase the success in outdoor environments, recent computer vision algorithms can be evaluated and adapted to the system, which may improve the overall performance. Considering the inherited issue with this approach, which requires the system to accept the first three videos as authentic, wireless infrastructure or beacon style solutions can be integrated into this so that it will act as a failsafe and provisioning solution. Hence, the existing system will be merged with a network-based approach, and their findings will be shared with academia.

ACKNOWLEDGEMENT

The authors declare that there was no conflict of interest in the course of this study.

REFERENCES

[1] Saroiu, S., & Wolman, A. (2009, February). Enabling new mobile applications with location proofs. In Proceedings of the 10th workshop on Mobile Computing Systems and Applications, 1-6.

[2] Ferreira, J., & Pardal, M. L. (2018, November). Witness-based location proofs for mobile devices. In 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), IEEE, 1-4.

[3] Kenan, M. (2021, April). Comparative analysis of localization techniques used in lbs. In 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), IEEE, 300-304.

[4] Martins, P., Abbasi, M., Sa, F., Celiclio, J., Morgado, F., & Caldeira, F. (2019). Intelligent beacon location and fingerprinting. Procedia Computer Science, 151, 9-16.

[5] Alamleh, H., & AlQahtani, A. A. S. (2020, July). A cheat-proof system to validate GPS location data. In 2020 IEEE International Conference on Electro Information Technology (EIT), IEEE, 190-193.

[6] Saroiu, S., and Wolman, A. (2019, February). Enabling new mobile applications with location proofs. In Proceedings of the 10th workshop on Mobile Computing Systems and Applications, 1-6.

[7] Wang, X., Pande, A., Zhu, J., and Mohapatra, P. (2016). STAMP: Enabling privacy-preserving location proofs for mobile users. IEEE/ACM transactions on networking, 24(6), 3276-3289.

[8] Zhu, Z., and Cao, G. (2011, April). Applaus: A privacy-preserving location proof updating system for location-based services. In 2011 Proceedings IEEE INFOCOM, IEEE, 1889-1897.

[9] Javali, C., Revadigar, G., Rasmussen, K. B., Hu, W., and Jha, S. (2016, November). I am alice, i was in wonderland: secure location proof generation and verification protocol. In 2016 IEEE 41st conference on local computer networks (LCN), IEEE, 477-485.

[10] Davis, B., Chen, H., and Franklin, M. (2012, May). Privacy-preserving alibi systems. In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, 34-35.

[11] Talasila, M., Curtmola, R., and Borcea, C. (2010, December). Link: Location verification through immediate neighbors knowledge. In International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services, Springer, Berlin, Heidelberg, 210-223.

[12] Gambs, S., Killijian, M. O., Roy, M., and Traoré, M. (2014, October). PROPS: A privacy-preserving location proof system. In 2014 IEEE 33rd International Symposium on Reliable Distributed Systems, IEEE, 1-10.

[13] Hasan, R., and Burns, R. (2011). Where have you been? secure location provenance for mobile devices. arXiv preprint arXiv:1107.1821.

[14] Logan, L., Davids, C., & Davids, C. (2020, May). Determining the Indoor Location of an Emergency Caller in a Multi-story Building. In 2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR), IEEE, 1-6.

[15] Goh, B. S., Mahamad, A. K., Saon, S., Isa, K., Ameen, H. A., Ahmadon, M. A., & Yamaguchi, S. (2020, January). IoT based indoor locating system (ILS) using bluetooth low energy (BLE). In 2020 IEEE international conference on consumer electronics (ICCE), IEEE, 1-4.

[16] Nosouhi, M. R., Yu, S., Zhou, W., Grobler, M., and Keshtiar, H. (2020). Blockchain for secure location verification. Journal of Parallel and Distributed Computing, 136, 40-51.

[17] Nosouhi, M. R., Sood, K., Yu, S., Grobler, M., and Zhang, J. (2020). PASPORT: A secure and private location proof generation and verification framework. IEEE Transactions on Computational Social Systems, 7(2), 293-307.

[18] Barabas, P., Regnath, E., & Steinhorst, S. (2020). COLAW: Cooperative Location Proof Architecture for VANETs based on Witnessing. In 2020 International Conference on Omni-layer Intelligent Systems (COINS), IEEE, 1-8.

[19] Zafar, F., Khan, A., Malik, S. U. R., Ahmed, M., Maple, C., and Anjum, A. (2021). MobChain: Three-way collusion resistance in witness-oriented location proof systems using distributed consensus. Sensors, 21(15), 5096.

[20] Gulliver, S. R., Serif, T., and Ghinea, G. (2004). Pervasive and standalone computing: the perceptual effects of variable multimedia quality. International journal of human-computer studies, 60(5-6), 640-665.

[21] Bucher, D., Rudi, D., and Buffat, R. (2018, January). Captcha your location proof—A novel method for passive location proofs in adversarial environments. In LBS 2018: 14th International Conference on Location Based Services, Springer, Cham, 269-291.

[22] Maia, G. A., and Pardal, M. L. (2019). CROSS: loCation pROof techniqueS for consumer mobile applicationS. INForum. Guimaraes, Portugal.

[23] Santos, H. F., Claro, R. L., Rocha, L. S., and Pardal, M. L. (2020, October). STOP: A Location Spoofing Resistant Vehicle Inspection System. In International Conference on Ad-Hoc Networks and Wireless, Springer, Cham, 100-113.

[24] Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., and Kompatsiaris, I. (2019). Visil: Fine-grained spatio-temporal video similarity learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 6351-6360.

[25] (2022, May). WebRTC 1.0: Real-Time Communication Between Browsers. https://w3c.github.io/webrtc-pc/

[26] Suciu, G., Stefanescu, S., Beceanu, C., and Ceaparu, M. (2020, June). WebRTC role in real-time communication and video conferencing. In 2020 Global Internet of Things Summit (GIoTS), IEEE, 1-6.

[27] Rhinow, F., Veloso, P. P., Puyelo, C., Barrett, S., and Nuallain, E. O. (2014, January). P2P live video streaming in WebRTC. In 2014 World Congress on Computer Applications and Information Systems (WCCAIS), IEEE, 1-6.

[28] https://www.w3.org/(2022, May 5). World Wide Web Consortium (W3C).

[29] Koh, E. (2010, November). Conferencing room for telepresence with remote participants. In Proceedings of the 16th ACM international conference on Supporting group work, 309-310.