Captcha Your Location Proof—A Novel Method for Passive Location Proofs in Adversarial Environments

Dominik Bucher, David Rudi and René Buffat

Abstract A large number of online rating and review platforms allow users to exchange their experiences with products and locations. These platforms need to implement appropriate mechanisms to counter malicious content, such as contributions which aim at either wrongly accrediting or discrediting some product or location. For ratings and reviews of locations, the aim of such a mechanism is to ensure that a user actually was at said location, and did not simply post a review from another, arbitrary location. Existing solutions usually require a costly infrastructure, need proof witnesses to be co-located with users, or suggest schemes such as users taking pictures of themselves at the location of interest. This paper introduces a method for location proofs based on visual features and image recognition, which is cheap to implement yet provides a high degree of security and tamper-resistance without placing a large burden on the user.

1 Introduction

In recent years the impact of online ratings and reviews on the decisions people make has steadily risen, and has thus also moved into the focus of research. Among others, the empirical analysis by Ye et al. (2011) showed that a 10% increase in the online rating of hotels led to a boost of online bookings by more than 5%, while Anderson et al. (2012) found that an extra half star rating causes restaurant sales to go up by 19%. As a consequence, the number of corresponding platforms has risen and ratings and reviews have become an important factor for business owners.

D. Bucher (✉) · D. Rudi · R. Buffat
Institute of Cartography and Geoinformation, ETH Zurich, Stefano-Franscini-Platz 5, 8093 Zurich, Switzerland
e-mail: dobucher@ethz.ch

D. Rudi
e-mail: davidrudi@ethz.ch

R. Buffat
e-mail: rbuffat@ethz.ch

P. Kiefer et al. (eds.), *Progress in Location Based Services 2018*, Lecture Notes in Geoinformation and Cartography, https://doi.org/10.1007/978-3-319-71470-7_14

Generally, online ratings and reviews (in the following only referred to as reviews) can be separated into those that refer to a product and those that refer to a point of interest (POI), such as a hotel or a restaurant. This work focuses on the latter. While some platforms are specialized on certain domains, such as restaurants or hotels (as in the case of TripAdvisor[1]), general purpose platforms such as Yelp[2] or Google Maps[3] also exist. Using these platforms, it is possible to review any POI from a train station to a national park.

As online reviews directly influence the revenue of businesses, the incentive to create fake reviews is high. Not surprisingly, Hu et al. (2011) were able to detect manipulations in online reviews on both Amazon[4] and Barnes & Nobles.[5] Currently the major platforms do not implement any sophisticated measures to prevent the creation of fake reviews, other than that reviews need to be written by humans. This makes it easy to create malicious reviews, such as fake positive reviews for the own business or fake negative reviews for the competitors. Thanks to crowdsourcing platforms such as Mechanical Turk,[6] companies that offer the creation of fake reviews have access to a large human workforce. As a consequence, the state of the art systems are not able to efficiently prevent the creation of fake reviews.

Hence, in order to prevent fake reviews, it is not only necessary to check if a user is human, but also to verify that she visited the location she wishes to review. Such a "proof of location" or "location proof" can for example be achieved by letting the user solve a location based challenge. To do so, the location based challenge needs to fulfill the following properties:

- The challenge should only be solvable if the user is present at the correct location.
- It should not be possible to use a solved challenge a second time.
- A solution to the challenge should only work for one particular location.
- Business owners or other entities with positive or negative intentions with regard to the location of the challenge should not have any influence on the challenge.
- In order to ensure scalability there should be no need to locally install additional hardware or to require other users or entities to be present at the same time.

The first three properties are obviously required in order to ensure that a user is at a specific location. For example, if the challenge would simply consist of a secret code that is attached to the wall of a restaurant, this key could easily be distributed to other users. Furthermore, the business owner can choose whom he wants to show the key, thus avoiding potential bad reviews.

In this work, we propose a location based challenge using photographs. A user must take a picture of a location with her smartphone camera in order to prove that

[1]https://www.tripadvisor.com.

[2]https://www.yelp.com.

[3]https://www.maps.google.com.

[4]https://www.amazon.com.

[5]https://www.barnesandnoble.com.

[6]https://www.mturk.com/mturk/welcome.

she actually is at that location. Specifically, the user has to take a picture that overlaps by 50% with the right part of an existing picture. The verification is achieved by matching the photo to the previous photos. As nowadays nearly everybody has a smartphone with a decent camera, no additional infrastructure is required. Matching of photos is a well-studied problem. State-of-the art algorithm can automatically match multiple images to create panoramic images (Brown and Lowe 2007). These algorithms work by first extracting and matching scale-invariant feature transform (SIFT) features (Lowe 2004) from the different images. Thus, these algorithms inherently must decide if two images can be matched or not. Even when having a large database of existing photos, it is unlikely that an existing photo, not yet known to the system, matches a previously taken photo.

This paper is structured as follows. The next section discusses the relevant literature. We then present our method followed by an analysis of the adversarial model, as well as a simulation thereof. Finally, we discuss the location proof and draw our conclusions.

2 Related Work

A large body of work treats location proofs and secure location claims, as proving that a device or person is at a claimed position is an important step in many tasks. Location-based access control and authentication (Sastry et al. 2003; Francillon et al. 2011), interaction with online location-based services (Zhu and Cao 2011; Javali et al. 2015; Khan et al. 2014), or people-centric sensing (with smartphones) (Talasila et al. 2013) are examples where adversarial users might want to fake their locations in order to gain access, get additional benefits from services, or simply disturb the system. Our motivating examples are *online reviews*, where users can post their experiences with a service or at a location to a central system, making it available for other people which might be thinking about using the same service, or visiting the same location. In such settings, it is not uncommon for the different entities to cheat by posting fake reviews about their own service or a competitors' one (Mayzlin et al. 2014). Mayzlin et al. (2014) propose a methodology for detecting review manipulations, and find examples of both positive as well as negative manipulation on different review platforms, in particular when a competing service is located closely to the one being reviewed. Optimally, users would have to prove that they a) actually were or currently are at the location they are reviewing, and b) that they are not owning the business, nor being otherwise closely affiliated with it.

In general, such proofs require an active component, i.e., a device that has both computational and communicative powers, and a specialized communication protocol between the device and a user's smartphone. For example, Sastry et al. (2003) present the Echo protocol, which allows a set of *verifiers* V to verify that a *prover* p is in a certain *region of interest* R. Echo requires the verifier and prover to be able to communicate both using radio frequency as well as sound – in its simplest form the prover echos a request by the verifier using ultrasound, and the time required for this action bounds the maximal distance the prover is from the verifier. Ultimately, such *time-of-flight* or *time-difference-of-arrival* based approaches are very common for location proofs (Brands and Chaum 1994; Waters and Felten 2003), but require specialized devices at each location, which makes them an unfavorable choice for review platforms. In addition, to bound the round trip time (from verifier to prover and back), the prover has to be able to compute a response within a short time frame, requiring a device capable of doing so. This also implies that the active device is in control or possession of the reviewee, which is usually not given for online reviews, where the central platform has no direct connection to the individual services and locations. In the past, different communication technologies, such as Wi-Fi (Waters and Felten 2003; Luo and Hengartner 2010; Saroiu and Wolman 2009; Sastry et al. 2003; Javali et al. 2016), Bluetooth, (Mengjun et al. 2016; Wang et al. 2016; Zhu and Cao 2011) or RFID (Gao et al. 2012) were used for *time-of-flight* location proofs.

Another line of research concerns location verification utilizing third-party witnesses. Khan et al. (2014) require a spatio-temporally co-located entity to be present when the verifier tries to verify the provers location claim. Witnesses register with a location authority, and are used to generate location proofs (which are sent to the prover), which in the end can be presented to another party requiring location verification. Their protocol is resistant to malicious verifiers, provers, and witnesses, but requires the presence of even more active devices. Similar methods requiring witnesses are described in literature (Mengjun et al. 2016; Wang et al. 2016). While finding a witness might be possible for online reviews, it again is difficult for the review system to control location authorities at every location.

In our work, we propose a weaker form of location verification, inspired by so-called Captchas (Von Ahn et al. 2003). In essence, "a Captcha is a cryptographic protocol whose underlying hardness assumption is based on an [artificial intelligence (AI)] problem" (Von Ahn et al. 2003, p. 296). Commonly, Captchas are known from registration websites, where they appear as distorted images, in which a user has to recognize letters or numbers in order to verify his or her *human* nature. Saroiu and Wolman (2009) present an approach inspired by Captchas, which has the intention of proving that a person (i.e., not a device) is at a certain location. For this, they require that a picture of the person at the location is being sent to the verifier. The proof is made stronger by making the person hold up a paper, on which a certain requested message by the verifier is written. This prevents a malicious user from simply reusing the same picture over and over again. Our approach differs from the approach in Saroiu and Wolman (2009) in that we do not require the person to be present in the picture (i.e., we verify the location of the device), and that the repeated use of the same picture is prevented by requesting a picture at a random (but well-defined) location. It also differs from conventional Captchas, because its underlying hardness assumption is based on location properties, i.e., only someone present at the location is able to easily solve the problem.

3 Method

This section introduces the “Captcha your Location Proof” (CLP) method. The CLP method works with little to no infrastructural overhead, and has a similar mechanism to that known from Captchas. In particular, CLP uses the smartphones’ capability of taking pictures for proving a client’s location. The section starts with presenting the stakeholders, i.e., the natural persons or organizations involved in the execution of the proof. Then, the actual procedure describing both the underlying data structure and the communication between a client and a location based service provider employing the location proof method is explained in detail.

3.1 Stakeholders

The CLP approach involves three different stakeholders, the location based service provider, the location owner and the client.

Location Based Service Provider. The *location based service provider* (called: provider) is an organization or a group which wants to use CLP to ensure that clients of its (crowd sourced) services receive authentic information about locations. In particular, the provider acts as a trusted entity throughout this process. The provider typically hosts CLP within a web application, such as an online review site.

Location Owner. The *location owner* is an organization or a group which manages a particular location of interest, such as a restaurant or a park. The location owner does not have to set up any infrastructure for CLP to work.

Client. The *client* (or also *user*) is both the consumer and producer of information regarding the locations made available through the provider.

3.2 Procedure

The Data Structure. For the CLP approach we focus on the client as a producer of information, i.e., someone who wants to add some kind of information (e.g., a rating, a description, etc.) to a particular location she previously visited (e.g., a restaurant, a park, etc.). The provider (technically represented by an appropriate IT infrastructure) requires an *image* of the location from the client as a proof of presence. We define the combination of these three data items (i.e., information, location and image) together with the client as a *contribution* (cf. Fig. 1).

The images of contributions of different users are required to overlap partially, i.e., new images need to partly depict scenes of already existing images. In our data structure, this means that each image contains a reference to its successor, forming a directed tree $\mathscr{I}_t$. A successor of an image i_l is an image i_k that overlaps with 50% of the right half of i_l (cf. Fig. 4). Images taken by different clients from the same

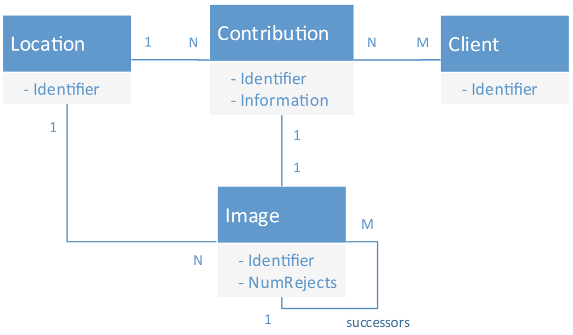

Fig. 1 The *simplified* class diagram for the data structure underlying the CLP approach

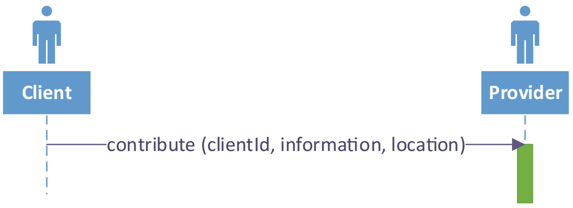

Fig. 2 An UML sequence diagram depicting the "The Contribution" step of the CLP procedure

perspective constitute the level l in the image tree $\mathscr{I}_t$. In particular, all images $i_{l,1}, \ldots, i_{l,n}$ at level l of the image tree have all images $i_{k,1}, \ldots, i_{k,m}$ of level k in the image tree as successors. Consequently, connecting the images of a path through the image tree would result in a big panoramic image representation of a location. We also store the number of times an image was rejected or accepted as a valid representation of a location.

Every now and then we ask clients to start a new image tree $\mathscr{I}_t$ for a particular location. Therefore, a location refers to one or more image trees $[\mathscr{I}_0, \ldots, \mathscr{I}_n]$ represented by their respective root image, and t indicates the tth tree in that list of trees.

The Contribution. The CLP approach starts with a *contribute* message sent from the client to the provider. The message contains the client's ID, the information she wants to contribute and the location the contribution is for (see also Fig. 2). For example, Alice wants to create a contribution for a restaurant she visited in Tokyo (see Fig. 3). She therefore sends the following (simplified) message to her provider: "(id: alice2018, location: (Tasty Edamame, Tokyo, Japan), information: ('5 stars', 'amazing sushi place'))".

The Challenge. After receiving the client's contribution request, the provider creates a temporary "Contribution" instance and returns its ID (for an appropriate communication tracking) together with a challenge (see also Fig. 5). That is, the provider challenges the user to prove she actually is at a location by taking a picture of it. More precisely, this is done by choosing from one of the following challenges, i.e., either to:

- *append* a picture to a known image tree of the location, i.e., taking a new picture that partially overlaps with an existing one in that tree, or to

Fig. 3 The location the "Alice" wants to create a contribution for

- *prove* an existing picture, i.e., taking a new picture that completely overlaps with a known picture in a known image tree, or to take a completely
- *new* picture, which is not compared to any other picture and constitutes the root for a new image tree.

Each of these challenges is chosen with a different probability. Formally speaking, the provider chooses the challenge $c \in \{append, prove, new\}$, where $p_{append} \in [0, 1], p_{prove} \in [0, 1]$ and $p_{new} \in [0, 1]$, with $p_{append} + p_{prove} + p_{new} = 1$.

To reduce the complexity, we assume that images can only be appended to the right of an existing image, i.e., there is at most one image per tree a user can append to, namely the rightmost one. We denote the list of all images for a location, i.e., the concatenation of all images of all image trees $\mathscr{I}_t$ as $I = [\mathscr{I}_0, \dots, \mathscr{I}_n]$. We further define an image $i \in I$ by $I[idx]$, where $idx = (idx_t, idx_l, idx_i)$ references a single tree by idx_t, a particular level by idx_l, and an image on this level by idx_i.

The image for an *append* challenge is defined as $i_c = I[idx]$, with $idx_l = |\mathscr{I}_t| - 1$ for a randomly chosen image tree $\mathscr{I}_t$ ($idx_t = t$; idx_i is randomly chosen from all the available images in the tree at that level). For the *prove* challenge the image is defined as $i_c = I[idx]$, where $idx_l < |\mathscr{I}_t| - 1$ for a chosen tree, i.e., the image is not the rightmost image in this image tree. Finally, to create a *new* image tree $\mathscr{I}_{n+1}$ the provider asks the client for an image and accepts whatever the client responds.

Note that, even though the provider actually chooses out of three different challenges, from the client's perspective there is no actual difference between *append* and *prove*, since the provider always sends some image i_c and marks 50% to the right of it for matching or asks her to take a completely new picture. If we assume that each user can only make one contribution for any location, the request is rejected if such a contribution has already been made.

In our example from Alice's perspective the provider responds to her depending on whether she is the first or *n*th contributor for the location '(Tasty Sushi, Tokyo, Japan)':

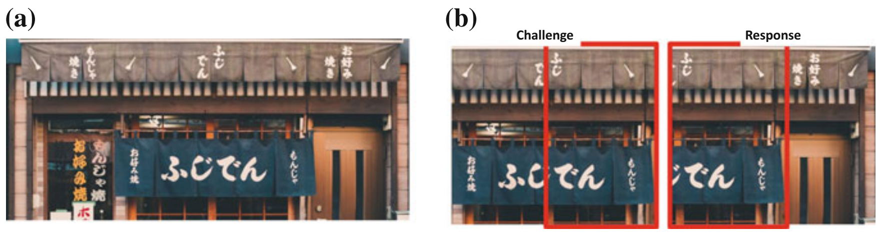

Fig. 4 **a** The picture Alice could take given the *new* challenge. **b** After receiving the *append* challenge with the picture to the left i_c, Alice could take the picture to the right i_r to match the right hand side (red boundary) of it

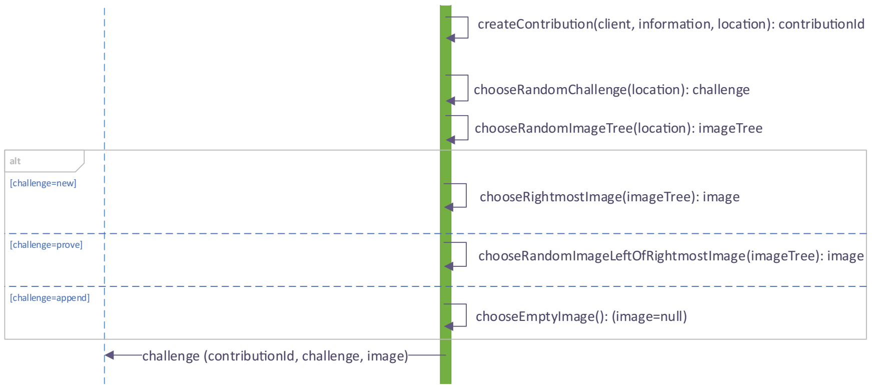
createContribution(client, information, location): contributionId
chooseRandomChallenge(location): challenge
chooseRandomImageTree(location): imageTree
alt
[challenge=new]
chooseRightmostImage(imageTree): image
[challenge=prove]
chooseRandomImageLeftOfRightmostImage(imageTree): image
[challenge=append]
chooseEmptyImage(): (image=null)
challenge (contributionId, challenge, image)

Fig. 5 An UML sequence diagram depicting the “The Challenge” step of the CLP procedure

- *In case of the first contribution* Alice receives the response (contributionId: 42, challenge: ‘new’, image: null) and is thus asked to take a picture of the “Tasty Sushi” (see Fig. 4a).
- *In case of the* nth *contribution* Alice either receives the response (contributionId: 42, challenge: ‘new’, image: null) and is again asked to take a picture of the “Tasty Sushi” (see Fig. 4a), or she receives the response (contributionId: 42, challenge: ‘append’, image: i_c) and is asked to append to the right of an existing picture (see Fig. 4b).

The Response. After receiving the challenge, the client responds with either one of the following (see also Fig. 6):

- *img*, i.e., taking the picture, or
- *reject*, i.e., not taking the picture (except if we had a *new* challenge)

Rejecting to send an image could be either because the client cannot or does not want to respond to the challenge.

In case of Alice this means that she could either:

- *Take a picture* and respond with (contributionId: 42, image: i_r) (see also Fig. 4b), or

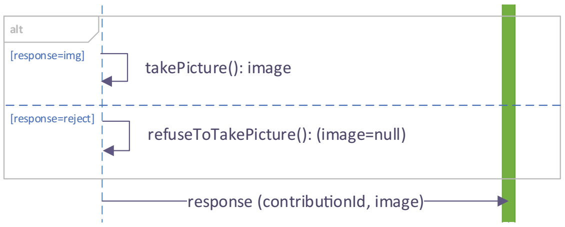
alt
[response=img]
takePicture(): image
[response=reject]
refuseToTakePicture(): (image=null)
response (contributionId, image)

Fig. 6 An UML sequence diagram depicting the "The Response" step of the CLP procedure

- *Reject* i_c and respond with (contributionId: 42, image: null)

The Reaction. The provider's reaction to the the client's response depends on the previously chosen challenge (see also Fig. 7):

- if the provider chooses an *append* challenge with the image i_c
  - and the client responded with *img* and the image i_r: the provider checks whether there is a 50% overlap between i_c and i_r. If not, the client's response is rejected and she is asked to submit a new image $i_{r,new}$. Otherwise, i_r is added to $\mathscr{I}_t$ at level $|\mathscr{I}_t|$ and is defined as successor to all images at level $|\mathscr{I}_t|-1$. The provider then responds with an "OK" message to the client.
  - and the client responded with *reject*: the provider increments the "NumRejects" counter of i_c and goes back to the **The Challenge** step and reruns the process from there on.
- if the provider chose a *prove* challenge with the image i_c
  - and the client responded with *img* and the image i_r: the provider checks for a 50% overlap between i_c and i_r, as well as for a 100% overlap with all images stored at the same level as i_c. If there was already no 50% overlap, the client's response is rejected and she is asked to submit a new image $i_{r,new}$. If however the 100% comparison fails, the provider accepts the contribution anyway, but increases the "NumRejects" counter for the images that do not match i_r. i_r is then added to $\mathscr{I}_t$ at the level of $i_c + 1$ and is defined as successor to all images at the level of i_c. The provider responds with an "OK" message to the client.
  - and the client responded with *reject*: the provider increments the "NumRejects" counter of i_c and all images at the same level in $\mathscr{I}_t$ that match it, and goes back to the **The Challenge** step and reruns the process from there on.
- if the provider chose a *new* challenge
  - and the client responded with *img* and the image i_r: the provider creates a new image tree $\mathscr{I}_{n+1}$ with i_r as the root image and n being the number of image trees for the location. The provider responds with an "OK" message to the client.

At any point in time, the client can cancel the procedure; this is not treated separately here.

In case of Alice this means that she could either:

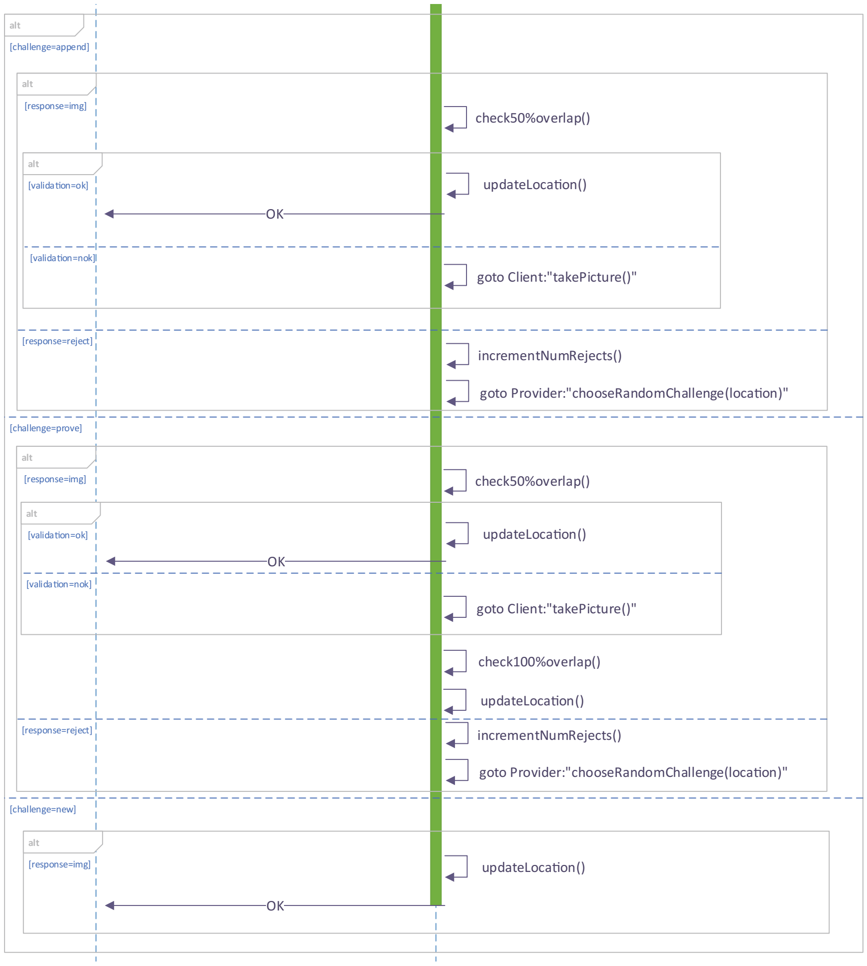

Fig. 7 An UML sequence diagram depicting the "The Reaction" step of the CLP procedure

- *Take a picture* and respond with (contributionId: 42, image: i_r), or
- *Reject* i_c and respond with (contributionId: 42, image: null)

3.3 Adversarial Models

3.3.1 Assumptions

We assume that the service provider has a crowd sourced location based service. In particular, the service provider wishes to ensure that its clients' contributions are authentic, i.e., that the client actually was at the location she made the contribution for. The communication between the clients and the server is secure, and the server (and thus also the service provider) is a trusted entity. We assume that the location owner wants the information which is made available to the clients by the service provider to be of a positive nature.

Furthermore, the clients, to prove the authenticity of their contribution, use a mobile phone to take a picture at the locations they wish to contribute information about. The picture along with their current (GPS) position, as well as a set of any other information (e.g., a rating) is then sent to the service provider. Moreover, the clients trust the service provider to ensure that the location based information they receive from other clients is authentic. The image processing is assumed to be able to detect if two images are similar or completely distinct. In addition, we assume the challenge images to be watermarked, i.e., it is not possible for an adversary to simply reuse these images at a later point in time.

Finally, we we will assume scenarios, where adversarial contributions appear in "bursts", i.e., ignoring a location's actual quality, adversarial contributions will either occur at the beginning or the end of a contribution "history". This is based on the observations from Hu et al. (2011), i.e., an adversary might either create novel contributions for a location that has none, or try to negate whatever contributions already exists. In either case the contributions aim at either discrediting a location or boosting its reputation. To realize this, they have to occur within a certain clustered time-frame. This holds true even if adversaries would try to make the contributions with temporary displacements to avoid attracting attention.

3.3.2 Threats

Threats can originate from both clients and location owners. Reasons to manipulate contributions could be that the client is a competitor of the location owner and wishes to discredit the location, or that she is employed by the location owner to add positive information regarding the location. The location owner could also act as a client herself.

Generally, an adversary (either a client or location owner) can easily spoof her GPS location. That step would by itself however not necessarily break the CLP approach. In particular, we identified the following additional actions that are necessary to pose a threat:

- **Malicious first contribution**. An adversary (client or location owner) can contribute with an initial image which is not of the location L she claims to be at. For example, she can take a picture of any other location L' and have that image uploaded. Afterwards the adversary can add further images and information from the fake initial location L'.
- **Similar looking locations**. An adversary might try to make a contribution from a location L' for the location L by trying to find an image that she believes might be accepted by the system. That is, after receiving the challenge for a segment of

an image of L, she searches an image database and finds one that she believes has similar enough features.

- **Catalog of real images**. An adversary could take many pictures of the actual location L and store those in a data storage. Afterwards the adversary could create contributions and use the image database for looking up an appropriate proof that complies with the CLP approach.
- **Systematically boosting fake contributions**. Adversaries could systematically reject all images they know are real and try to find images fitting to fake images.
- **Creating a fake image tree**. Adversaries could systematically reject all images until they receive the *new* challenge and afterwards continue rejecting any image that is not within that image tree and only append to that tree.
- **Accepting only new challenges**. Adversaries could systematically reject all challenges until they receive the *new* challenge and create only new image trees with fake images.
- **Man-in-the-middle**. A known attack against the Captcha system was an application that relayed Captchas to third users, which were incentivized by various rewards to return the solved problem to the original user. With our system, such man-in-the middle attacks are much harder as somebody needs to be present at the actual location.

The next section will evaluate these threats using a simulation approach, and discuss countermeasures for the threats which CLP cannot prevent.

4 Technical Evaluation

In this section, we will discuss the advantages and disadvantages, as well as the presence of adversaries in the CLP approach in more depth. Recall that upon receiving an image, a user has two options to respond: $\{img, reject\}$, where the first response means sending an *image* i (either appended to the *challenge image* i_c with an overlap of 50%, or a completely new one), and the second means to *reject* the challenge image, thus discrediting it, and receiving a new challenge (this second response is not available for the *new* challenge).

4.1 Honest and Adversarial User Responses

For each challenge, we have to evaluate the possible answers of a user u_h (honest), and a user u_a (adversarial). Note that we do not make any statements on the honesty of the contribution of a user, but only on its location—someone who is present at a certain location can still send reviews discrediting a competitor. The user has to be at the location, though, and cannot simply outsource the task to some company operating from another location.

The *new* challenge. If a user receives a *new* challenge, there is only one possible response, namely to send a new image i, which will be stored in the CLP system as part of a user's contribution. As we do not know if u is honest or adversarial, the system simply stores the image without any further processing.

The *append* challenge. When appending to an image i_c, honest and adversarial users react differently. In the following, we assume that u_h always is correctly following protocol, while u_a chooses either the potentially best outcome for herself, or the worst impact on the system (to destabilize it). As such, user u_h will perform one of the following two actions:

- The user finds the scene depicted by the challenge image i_c at the location, and is able to append as requested. She will send the appended image i, which will be stored as part of a contribution associated with the given location.
- The user is not able to find the scene depicted by i_c, and thus has to *reject* the challenge. The challenge image i_c is now discredited, i.e., the CLP system increases the rejection count of image i_c. This will later be important, when we assess which users to trust, and which users to flag as adversarial.

A dishonest user u_a will react completely different upon receiving an *append* challenge. Let us first assume the challenge image i_c is honest, i.e., the scene depicted by it can be found at the location. In this case, the adversarial user has two options:

- She can reject the honest image i_c, thus discrediting it, and receive a new challenge. The CLP provider will store that u_a discredited the image, and continue as usual.
- Append a dishonest image by taking the overlapping half of i_c, generating some arbitrary image for the other half, and sending it back to the CLP provider. Again, the CLP system does not know this is an adversarial image yet, so it will store the the image as part of the user's contribution.

Choosing the first option, u_a builds up distrust towards honest users, which can be an advantage, as long as her adversarial identity is not revealed. With the second option, u_a builds up trust, as long as the contribution is not discredited by honest users. In case the challenge image i_c itself is dishonest, u_a would have the same choices again. However, rejecting a dishonest image would discredit the adversarial owner of that image, and result in uncovering the owner's mischievous doings. As such, it would always benefit honest users, and the trustworthiness of the system. We can thus assume that in order to get the maximal benefit for herself, a malicious user always appends to a dishonest image.

We introduce the probabilities p_r (reject) and $p_a = 1 - p_r$ (append) with which the response of an adversarial user is chosen. As discussed above, $p_a = 1$ for a dishonest challenge image, as the adversary will always append. We will evaluate different values of p_r and p_a for a honest challenge image below, but for example, if the probability to reject is $p_r = 1$, this means the adversarial user will reject until she either has to *append* to a dishonest image or gets a challenge for a *new* image (the *prove* challenge will also always be rejected, as from a user perspective, *append* and *prove* look equal).

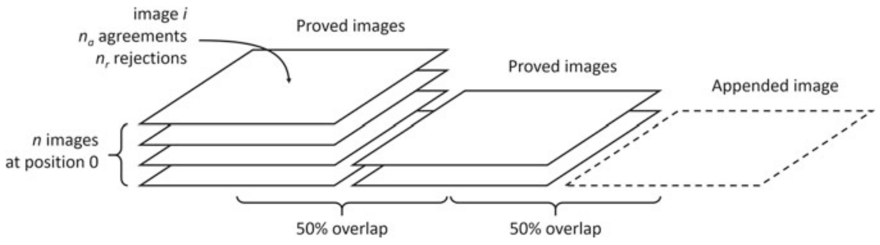

Fig. 8 Schematic representation of the images in the CLP system. Each image can stem from a honest or from an adversarial user, and has a certain number of agreements (successfully appended) and rejections associated with it (number of times the image was rejected during an *append* or *prove* challenge). Whenever a *new* challenge is responded with a new image, another such a "list of images" is started

The *prove* challenge. From a user's perspective, the *prove* challenge cannot be distinguished from the *append* challenge. As such, the responses are the same, but since the CLP system has additional knowledge this results in a different procedure on the system.

Namely, the system will check for any appended image, if it corresponds to any of the other images taken at this position. This results in a number of agreeing images, and a number of disagreeing ones. In the worst case, all adversaries work together, i.e., they send images that agree with each other, but not with the images of the honest users. On the other hand, all images of honest users always agree with each other, per our definition of honesty. Figure 8 shows the images associated with a given tree of images, displayed as a list of stacks of proved images. Such a list is started whenever an image is entering the system as response to a *new* challenge. The images in this list can stem from honest and adversarial users, and thus create a number of *trust votes* for each other. For example, in a stack with 5 honest images, and 3 adversarial ones, each of the honest images has 4 votes of trust, and 3 votes of distrust, while each of the adversarial images has 5 votes of distrust, and 2 votes of trust. The votes are simply computed by applying image processing techniques to all images, and measuring how many of the features of two images agree with each other. Note that these images would all overlap each other, for clarity they were drawn separated in Fig. 8.

In addition to these trust votes, each image can be appended to or be rejected in response to *append* and *prove* challenges, which is counted for each image in variables n_a (number of agreements) and n_r (number of rejections).

4.2 Assessing Adversarial Users

We are ultimately interested in determining which users are adversaries, and which users can be trusted. As a system, our initial trust towards each user is the same, but given user votes, we can determine which users we can trust more, and as a result, which users' reviews we have to remove from the system again. We can do this using the knowledge gained from rejected images, as well as from proved images that do not match.

In the following analysis, we restrict ourselves to saying that every user can at most prove her location once (i.e., each user is unique and does not appear for multiple images nor multiple locations). We reason that security improves if users are allowed to prove their location several times and at multiple locations, as honest users can build up trust, which can be weighted more when wanting to detect adversaries. Insofar, the situation considered here is a worst-case scenario which should improve in real-world systems.

As described above, we have two sets of votes for each image, one from rejections, and one from comparison to other proved images. The rejection votes make statements about users thinking that a particular image is adversarial (i.e., it will contain an arbitrary number of votes, depending on the number of times the image was chosen to be part of a challenge). The prove votes make a similar statement implicitly, and thus always contains a number of votes corresponding to all other proved images at the same location.

We can now count the positive votes in a set of prove images I_p (which should depict the same location) for any given image i (and thus the user who posted it) as:

$$v_{i,p} = n_a + agree(i, I_p)$$

where n_a is the number of users who successfully appended to this image and $agree(i, I_p)$ is the number of images of the same location that agree (i.e., have enough matching features for the image processing to recognize them as the same location) with this image. The number of negative votes is computed as:

$$v_{i,n} = n_r + (|I_p| - agree(i, I_p))$$

where n_r is the number of users who rejected to append to this image and $|I_p| - agree(i, I_p)$ is the number of images at the same place which do not agree with this image. We now trust an image (and thus the user who posted it) if $v_{i,p} > v_{i,n}$. This also means that we trust the location of the user to be genuine. Note that with this scheme, we can discover adversarial users at any later point in time, and eventually remove their mischievous reviews. We thus have to recompute trust whenever a new user contribution enters the system, and assess which users we trust, and whom we have to classify as adversarial.

4.3 Simulating CLP Challenges and User Responses

To evaluate the influence of different probabilities and parameters on the number of adversaries CLP is able to identify, we now present a simulation of the approach. The simulation model exactly follows the above described method, i.e., honest and adversarial users want to add a contribution to a system, and are challenged to either provide a *new* image, *append* to an existing one, or *prove* an image. While the *append* and *prove* challenges look equal from a user's point of view, the system handles their responses differently. Within the simulation, we model a restricted view of the world for the CLP system, as well as for the clients themselves. As the simulator knows everything (in particular, which users are honest, and which ones are adversarial), we can easily compute how many users were correctly classified as adversarial by the system, and how many honest users are wrongly accused of being dishonest. In an optimal system, no honest user would be treated as an dishonest, and no adversary would be treated as honest.

The simulation routine is shown in Algorithm 1. The functions *generateUser* (L1) and *generateChallenge* (L4) simply generate either *honest* or *adversarial* users, and *new*, *append* or *prove* challenges, according to the rules defined above. *insertImage* (L10) increases the agreement count, and inserts the image at the correct position in the right image tree. The core function *simulateStep* first generates a random user and a random challenge (L13–15), and then applies the logic described above to the image collection, depending on the type of challenge. If a user rejects a certain challenge image, the *goto* (L22/26) statements cause the simulator to generate a new challenge, and restart the procedure. Finally, we count the number of distrusted honest users $h_{distr.}$ as the percentage of all honest users, for whom the majority vote yielded that they should not be trusted, and $a_{trust.}$ as the percentage of all adversaries, for whom the vote yielded that they should be considered honest.

We ran the simulation for a range of values for the parameters p_{new}, p_{prov}, p_r and p_{adv}, in order to determine the best values for p_{new} and p_{prov} (p_{app} can be calculated from them), given different adversary strategies. Table 1 shows the best values (when minimizing a_{trust}) for p_{new} and p_{prov} for given probabilities of adversaries, and different adversarial strategies. *beg.* and *end* are two adversary strategies, where there either is a burst of adversaries in the beginning ($p_{adv} = 0.7$ for the first 25 contributions, and 0.1 for the rest) or in the end ($p_{adv} = 0.7$ for the last 25 contributions, otherwise 0.1). For each parameter combination, we ran the simulations five times for 100 contributions each, and measured the average of the final percentages for distrusted honest users and trusted adversaries.

Figure 9 gives exemplary outputs for three simulations. On the left, a scenario with $p_{adv} = 0.2$ is shown, while in the middle the adversaries are clustered in the

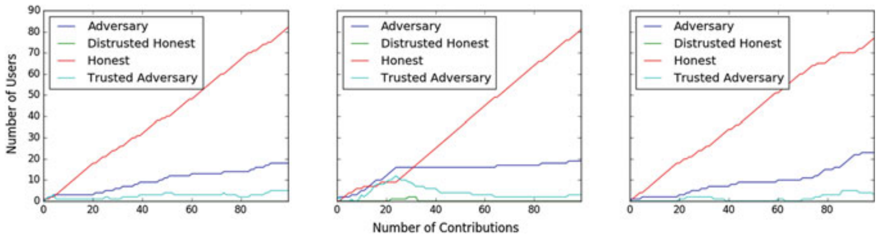

Fig. 9 Output of three simulation runs, where honest and adversarial persons use CLP to upload contributions to a review site

Algorithm 1: The simulator function, which is executed iteratively. Each iteration adds another user contribution, either from an honest or from an adversarial user. $\mathbf{I} = \{I_0, \dots, I_n\}$ corresponds to the image trees associated with a certain location, and is initially empty. The function *random()* generates a random number $r \in [0, 1]$, the function *randomImage(·)* selects a random image from all images in the trees passed to it, and the probabilities p_{adv}, p_{new}, p_{app}, and p_r describe the percentage of adversarial users, *new* challenges, *append* challenges, and the reject strategy of adversarial users. We assume *insertImage(·)* knows how to *append* a new image i_r to the image tree, given a predecessor i_c. A and R are accept and reject counters for all images, and are initially 0 for every image. Finally, *filter(·,f)* only returns the images which fulfill the predicate function f.

```
1  Function generateUser ()
2      if random() < $p_{adv}$ then return AdversarialUser
3      else return HonestUser
4  Function generateChallenge ($\mathbf{I}$)
5      if $|\mathbf{I}| = 0$ then return NewChallenge
6      $r \leftarrow$ random()
7      if $r < p_{new}$ then return NewChallenge
8      else if $r < p_{new} + p_{prov}$ then return ProveChallenge(randomImage($\mathbf{I}$))
9      else return AppendChallenge(randomImage($\mathbf{I}$))
10 Function insertImage ($i_c$, $i_r$)
11     $A[i_c] \leftarrow A[i_c] + 1$
12     append($\mathbf{I}$, $i_c$, $i_r$)
13 Function simulateStep ()
14     $u \leftarrow$ generateUser()
15     $c \leftarrow$ generateChallenge($\mathbf{I}$)
16     switch $c$ do
17         case NewChallenge do
18             $\mathbf{I} \leftarrow \mathbf{I} \cup \{I(i_r)\}$        ▷ Where $I(i_r)$ starts a new image tree.
19         case AppendChallenge($i_c$) or ProveChallenge($i_c$) do
20             if typeof $u$ = HonestUser then
21                 if typeofowner $i_c$ = HonestUser then insertImage($i_c$, $i_r$)
22                 else $R[i_c] \leftarrow R[i_c] + 1$; go to 15
23             else
24                 if typeofowner $i_c$ = DishonestUser then insertImage($i_c$, $i_r$)
25                 else
26                     if random() < $p_r$ then $R[i_c] \leftarrow R[i_c] + 1$; go to 15
27                     else insertImage($i_c$, $i_r$)
28     $h_{distrusted} \leftarrow |\text{filter}(\mathbf{I}, i \mapsto$ typeofowner $i$ = HonestUser and $v_{i,n} > v_{i,p})|$
29     $a_{trusted} \leftarrow |\text{filter}(\mathbf{I}, i \mapsto$ typeofowner $i$ = DishonestUser and $v_{i,n} < v_{i,p})|$
```

Table 1 Different scenarios, and the percentage of identified adversaries. $h_{distr.}$ is the percentage of honest users that are distrusted, $a_{trust.}$ is the percentage of adversaries that are wrongly trusted

p_{adv}	p_r	p_{new}	p_{prov}	$h_{distr.}$ [%]	$a_{trust.}$ [%]
0.1	0.2	0.05	0.35	0.0	1.7
	0.5	0.05	0.70	0.1	8.1
	0.8	0.05	0.05	3.3	9.5
0.2	0.1	0.05	0.25	0.0	4.8
	0.5	0.05	0.50	0.1	7.8
	0.8	0.05	0.25	1.1	16.0
0.4	0.1	0.05	0.50	6.7	27.4
	0.5	0.05	0.80	2.4	27.8
	0.8	0.05	0.80	7.6	42.4
0.8	0.1	0.10	0.35	15.3	87.5
	0.5	0.05	0.50	11.3	87.6
	0.8	0.05	0.30	14.1	89.4
beg.	0.1	0.05	0.20	0.4	14.2
	0.5	0.05	0.20	0.1	26.5
	0.8	0.05	0.75	0.4	33.1
end	0.1	0.05	0.85	0.4	13.6
	0.5	0.05	0.75	0.8	16.2
	0.8	0.05	0.40	0.7	24.4

beginning, and on the right in the end (as described in the previous paragraph). $p_{new} = 0.1$, $p_{prov} = 0.3$ and $p_r = 0.2$ were constant.

It can be seen from both Table 1 and Fig. 9 that CLP usually is able to identify a large number of adversaries. The best strategies for adversaries are to either simply suppress the honest users (large p_{adv}), or to choose a large p_r, i.e., simply reject everything until the are allowed to send a new image. However, we argue that such users could be identified (for example, by always sending a *append* images with many trust votes), and thus it is not a good choice for adversaries to choose p_r very large. In reality, we would hopefully also always see a substantial number of honest users (otherwise, the place would not be of interest), i.e., p_{adv} cannot get too large. Finally, it is interesting to see that p_{new} should be low. This is because *new* challenges are the primary means for adversaries to hide their malicious intent. In reality, we cannot chose p_{new} too small, as honest users must have a chance to eventually submit a genuine *new* picture (otherwise, they would always have to reject).

4.4 Countermeasures

As the previous section argued, CLP is inherently able to detect a substantial share of adversarial users. Attacks generally become more difficult with a growing number of images, because the number of possible segments that we could challenge the user with is equal to the number of images for the location. Thus, even when an attacker finds a similar location, assuming a high enough number of images exist, finding a match becomes more difficult. In particular, since we have the 100% check it is possible that the system chooses the same image multiple times countering the possibility of a similar looking image.

Another easy way to handle malicious contributions is to allow clients to request a reset of the contributions, i.e., removing all contributions for a location. Graduations of such a solution are possible too, e.g., removing only the last n contributions. In practice, this could be implemented by adding a trust or gamification (cf. Weiser et al. 2015) layer on top of CLP, which would allow "power users" to manually assess all images at a location in return for points or other game elements.

Additionally, it is possible to adapt the size and position of image segments. For example instead of choosing 50% to the right of an image, we could extract two snippets, one of size 200×200 pixels and one with 400×400 pixels at an randomly chosen position of the "stitched up" image and ask a client to take a picture containing the smaller segment. Afterwards we could extract a segment of 400×400 pixels from the response image and match that against the challenge image of same size.

Allowing users to make contributions for multiple locations would allow them to build up trust, which can be used to spot malicious users. This could be further enhanced by using time geography (Miller 2005) to assess whether a person could have traveled between two locations within a given time frame.

5 Discussion

In this work, we presented a method for posing a location based challenge that allows a location based service provider to verify if a user was as at a particular location. Such a location proof is particularly important in crowd-sourced scenarios where a service provider wishes to ensure a certain degree of authenticity of the collected information.

The underlying idea is simple: We ask a user to take a picture of the location she wants to make a contribution for. That picture needs to partially match an existing picture contributed by some other user. Should the image match, we store it in the system and compare it to an ever-increasing set of images depicting the same scene, which allows assessing whom we can trust, and whom we should classify as adversarial. The advantage of such a principle is that it can be applied in large scale, as nowadays nearly everybody has a smartphone with a decent camera and fast image processing algorithms exist. We thus avoid any additional infrastructure or contexts that are difficult to achieve, which current state of the art methods for location verification usually require (additional local infrastructure or co-located witnesses).

We thoroughly analyzed our method for possible attack scenarios and found that while we cannot ensure the authenticity of the location of every user, in many realistic scenarios CLP can detect the majority of attackers. That is, our method is not "bullet-proof" (like a cryptographic method) but makes malicious contributions in a real world scenario considerably more expensive (similar to Captchas, where the challenges are more difficult for automated systems, CLP challenges are more difficult for people absent from a certain location). Moreover, we were able to quantify how many attackers we can identify under which circumstances (in particular for which given probabilities). While our analysis is based on a simulation, the same technology could be used in a real (prototypical) system if supplied with data from a computer vision subsystem and integrated into a web application. As part of our work on location based need matching systems (Bucher et al. 2017), we are working on different location proof system implementations.

Nonetheless, the proposed method has certain limitations. The location challenge requires an honest user to be present at the location. This is a drawback regarding the convenience for the user. From a practical point of view the location of the photo of a location challenge needs to be identifiable by the user. This requires that this location is within the line-of-sight of the user. For normal locations such as restaurants this will not be a problem. However, POIs vary greatly in sizes, e.g., for a national park spanning multiple square kilometers finding the right location will not be feasible. This can be circumvented by also considering the GPS location of the user and only using images that are close to that GPS location. An additional challenge is changing POIs, for example when a restaurant is renovated or a fair moves to another location. However, one can argue that for these POIs the accuracy of an old review also potentially decreases very fast. Additionally, the countermeasure of resetting the contributions could be exploited here as well.

For practical deployment, gradual introductions of CLP are possible (as the hard requirement to take a matching picture could discourage people from contributing to a crowd-sourced service). For example, when using an "add pictures to your review" functionality, users could be asked to try to append to existing images. This could be transformed into a "trust score", which still allows anyone to contribute, but gives a higher weight to honest users. The CLP system can also be used to remove fake reviews long after they have been posted. For example, somebody could have created a large corpus of photos in advance and used them to create fake reviews. If one review was identified as a fake review, due to the graph structure of the matching photos, all connecting reviews can be found and removed from the system. Generally, even though our approach poses a comparably weak authentication proof, extending it with the many countermeasures we introduced and even combining it with other verification mechanisms can make it completely infeasible for adversaries to attack.

6 Conclusion and Future Work

This paper introduced the novel "Captcha your Location Proof" (CLP) approach, which in contrast to earlier research in the field of Location Proofs does neither rely on any additional infrastructure, nor on the client being in the picture. CLP rather relies on the principle that as long as there are more honest than dishonest clients, most of the adversarial ones can be detected and removed. Moreso, providers of location based services employing CLP can be sure that they are providing their clients with authentic contributions or reviews for locations or POIs. Furthermore, from an honest client's perspective the CLP method consists only of declaring the wish to create a review for a POI and taking a picture thereof. Together with the high availability of cameras in smartphones or even regular mobile phones, the simplicity of CLP makes it especially attractive for location based service providers such as review platforms.

Like all location proof approaches, CLP both strengths and weaknesses. We analyzed them in detail in our description of the adversarial model. In particular, we worked out a clear differentiation between honest and dishonest clients, as well as the possible threats they could pose. We then conducted a simulation of a possible real world application with predefined probability measures to demonstrate the behavior and practicability of CLP in practice. Finally, we presented a set of countermeasures that help increase the reliability of CLP and efficiently counter the threats. For future research it might be interesting to realize CLP with a real infrastructure and conduct user studies to both test the stability, practicability, as well as the user acceptance of the system in a real world environment.

Nonetheless, our simulation showed that the CLP approach as it was presented in this paper poses a scalable, easy to implement, easy to use, user friendly solution for location proofs in an adverserial environment.

Acknowledgements This research was supported by the Swiss National Science Foundation (SNF) within NRP 71 "Managing energy consumption" and by the Commission for Technology and Innovation (CTI) within the Swiss Competence Center for Energy Research (SCCER) Mobility and FURIES (Future Swiss Electrical Infrastructure).

References

Anderson M, Magruder J (2012) Learning from the crowd: regression discontinuity estimates of the effects of an online review database*. Econ J 122(563):957–989. http://dx.doi.org/10.1111/j.1468-0297.2012.02512.x

Brands S, Chaum D (1994) Distance-bounding protocols. Lect Notes Comput Sci 765:344–359

Brown M, Lowe DG (2007) Automatic panoramic image stitching using invariant features. Int J Comput Vis 74(1):59–73. https://doi.org/10.1007/s11263-006-0002-3

Bucher D, Scheider S, Raubal M (2017) A model and framework for matching complementary spatio-temporal needs. In: Proceedings of the 25th ACM SIGSPATIAL international conference on advances in geographic information systems. ACM

Francillon A, Danev B, Capkun S (2011) Relay attacks on passive keyless entry and start systems in modern cars. In: Proceedings of the 18th annual network and distributed system security symposium. the internet society. Citeseer

Gao H, Lewis RM, Li Q (2012) Location proof via passive RFID tags. Springer, Berlin, Heidelberg, pp 500–511

Hu N, Liu L, Sambamurthy V (2011) Fraud detection in online consumer reviews. Decis Support Syst 50(3):614–626. On quantitative methods for detection of financial fraud. http://www.sciencedirect.com/science/article/pii/S0167923610001363

Javali C, Revadigar G, Hu W, Jha S (2015) Poster: were you in the cafe yesterday?: location proof generation and verification for mobile users. In: Proceedings of the 13th ACM conference on embedded networked sensor systems. ACM, pp 429–430

Javali C, Revadigar G, Rasmussen KB, Hu W, Jha S (2016) I am Alice, I was in wonderland: secure location proof generation and verification protocol. In: 2016 IEEE 41st conference on local computer networks (LCN), Nov 2016, pp 477–485

Khan R, Zawoad S, Haque MM, Hasan R (2014) Who, when, and where? Location proof assertion for mobile devices. In: IFIP annual conference on data and applications security and privacy. Springer, pp 146–162

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

Luo W, Hengartner U (2010) Veriplace: a privacy-aware location proof architecture. In: Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems, GIS '10. ACM, New York, NY, USA, pp 23–32. http://doi.acm.org/10.1145/1869790.1869797

Mayzlin D, Dover Y, Chevalier J (2014) Promotional reviews: an empirical investigation of online review manipulation. Am Econ Rev 104(8):2421–2455

Mengjun L, Shubo L, Rui Z, Yongkai L, Jun W, Hui C (2016) Privacy-preserving distributed location proof generating system. China Commun 13(3):203–218

Miller HJ (2005) A measurement theory for time geography. Geogr Anal 37(1):17–45

Saroiu S, Wolman A (2009) Enabling new mobile applications with location proofs. In: Proceedings of the 10th workshop on mobile computing systems and applications, HotMobile '09. ACM, New York, NY, USA, pp 3:1–3:6. http://doi.acm.org/10.1145/1514411.1514414

Sastry N, Shankar U, Wagner D (2003) Secure verification of location claims. In: Proceedings of the 2nd ACM workshop on wireless security, WiSe '03. ACM, New York, NY, USA, pp 1–10. http://doi.acm.org/10.1145/941311.941313

Talasila M, Curtmola R, Borcea C (2013) Improving location reliability in crowd sensed data with minimal efforts. In: 2013 6th Joint IFIP wireless and mobile networking conference (WMNC). IEEE, pp 1–8

Von Ahn L, Blum M, Hopper NJ, Langford J (2003) Captcha: using hard AI problems for security. In: International conference on the theory and applications of cryptographic techniques. Springer, pp 294–311

Wang X, Pande A, Zhu J, Mohapatra P (2016) Stamp: enabling privacy-preserving location proofs for mobile users. IEEE/ACM Trans Netw 24(6):3276–3289

Waters B, Felten E (2003) Proving the location of tamper-resistant devices. Technical Report

Waters B, Felten E (2003) Secure, private proofs of location. Technical report

Weiser P, Bucher D, Cellina F, De Luca V (2015) A taxonomy of motivational affordances for meaningful gamified and persuasive technologies. In: Proceedings of the 3rd international conference on ICT for sustainability (ICT4S). Advances in computer science research, vol 22. Atlantis Press, Paris, pp 271–280

Ye Q, Law R, Gu B, Chen W (2011) The influence of user-generated content on traveler behavior: an empirical investigation on the effects of e-word-of-mouth to hotel online bookings. Comput Hum Behav 27(2):634–639. Web 2.0 in travel and tourism: empowering and changing the role of travelers. http://www.sciencedirect.com/science/article/pii/S0747563210000907

Zhu Z, Cao G (2011) Applaus: a privacy-preserving location proof updating system for location-based services. In: 2011 Proceedings IEEE INFOCOM, Apr 2011, pp 1889–1897