VerLoc: Verifiable Localization in Decentralized Systems

Katharina Kohls
Radboud University Nijmegen
kkohls@cs.ru.nl

Claudia Diaz
imec-COSIC KU Leuven
Nym Technologies SA
claudia.diaz@esat.kuleuven.be

Abstract

We tackle the challenge of reliably determining the geo-location of nodes in decentralized networks, considering adversarial settings and without depending on any trusted landmarks. In particular, we consider active adversaries that control a subset of nodes, announce false locations and strategically manipulate measurements. To address this problem we propose, implement and evaluate *VerLoc*, a system that allows verifying the claimed geo-locations of network nodes in a fully decentralized manner. *VerLoc* securely schedules roundtrip time (RTT) measurements between randomly chosen pairs of nodes. Trilateration is then applied to the set of measurements to verify claimed geo-locations. We evaluate *VerLoc* both with simulations and in the wild using a prototype implementation integrated in the Nym network (currently run by thousands of nodes). We find that *VerLoc* can localize nodes in the wild with a median error of 60 km, and that in attack simulations it is capable of detecting and filtering out adversarial timing manipulations for network setups with up to 20 % malicious nodes.

1 Introduction

Whenever network applications depend on specific locations for service nodes, they also depend on truthful location information [11, 51]. As GeoIP databases are not always reliable [18, 37, 44], active localization approaches that use timing measurements to derive geo-location have been proposed in prior work. Systems like Spotter [31], Octant [52], or constraint-based geo-location [21, 25, 30] send timing probes to targets from trusted landmarks that have a known location [13, 15]. Combining the timing measurements obtained by the set of landmarks allows to narrow down the location of the target. While this allows for predictions up to street-level granularity [12, 50], landmark-based systems rely on a trusted setup. Spotter and Co. depend on accurate ground truth information for landmark locations, as well as on honest accurate reporting of timing measurements by the landmarks. Such systems are neither robust to malicious landmarks that lie about their location or obtained measurements, nor to malicious targets that strategically manipulate timing measurements by, e. g., delaying responses to certain timing probes. This makes these solutions inadequate for decentralized settings that may be subject to adversarial conditions.

A scheme that allows to verify geo-location in networks in a fully decentralized manner – without relying on trusted landmarks or measurements – can be useful in a variety of scenarios. Here we highlight two use cases. First, *overlay anonymous communication networks* such as Tor [1] and Nym [2] route user connections through relays in multiple jurisdictions to protect against adversaries who have monitoring and coercion powers within a zone of adversarial control. Location diversity strengthens security, as it becomes harder to monitor, compromise or censor the full network [48]. Ensuring location diversity when routing a connection requires reliable geo-location information. However, prior work demonstrates that available solutions sometimes result in incorrect location information, e. g., 194 out of 6042 Tor relays were found to be in a different country than the one indicated by their GeoIP entry [27]. We can thus see that misleading location information is not only a theoretical possibility, and may even be actively used to obfuscate the whereabouts of network nodes. In addition to supporting geographic diversity, publicly verifiable node locations enable other functionalities dependent on accurate location data, such as location-aware anonymous routing policies that reduce end-to-end latency by favouring routes that travel a smaller distance [3, 40].

Second, location diversity is also important for resilience purposes in *peer-to-peer networks that jointly maintain a blockchain*, such as Bitcoin [3]. The concentration of network servers in certain geo-locations makes the network vulnerable to regional events, including natural disasters [54] as well as politically motivated interventions [8]. A method to reliably verify peer locations in a fully decentralized manner would enable such permissionless networks to incentivize location

[1] https://www.torproject.org
[2] https://nymtech.net
[3] https://bitcoin.org/

diversification, while ensuring that malicious peers cannot take advantage by faking their location. In particular, node locations can serve as one of the variables in the delegation criteria in systems based on delegated proof of stake [26].

These functionalities are compelling not just from an academic standpoint. The Nym network [14] has already integrated and deployed a prototype implementation of *VerLoc* that Nym mix nodes run twice a day. Nym wants to verify node locations to be able to enable in the future: (1) routing policy constraints to ensure routes traverse multiple jurisdictions and better protect from nation-state adversaries, (2) lower-latency location-aware routing, and (3) incentives for global location diversification via rewards and delegation of stake (e. g. premium reward rates for nodes located in geographical areas with lower node density). At the time of writing 3460 nodes have upgraded to the *VerLoc*-enabled version. We take advantage of this experimental prototype implementation to collect measurements and validate *VerLoc*'s performance in the wild.

Contribution. *VerLoc* tackles the challenge of verifying geolocations in a fully decentralized network without trusted authorities or landmarks, where up to 20 % of the network may be actively malicious. To do so, *VerLoc* uses a novel timing-based verification algorithm that is robust to network noise and can withstand strategic adversarial manipulation. *VerLoc* securely schedules Round Trip Time (RTT) measurements so that the adversary cannot influence randomized assignments. Based on pairwise RTT measurements, *VerLoc* applies trilateration to estimate the geo-location of nodes and verify their claimed whereabouts. *VerLoc* uses a broadcast channel to enable all nodes to share the information needed to verify all geo-locations. The measurement overhead involves sending a few thousand pings, and it remains constant as the network grows, while the data processing and storage overhead grows linearly with the number of nodes (by 200 B per node).

As preliminary step, we conduct an empirical study to derive a realistic network propagation model that accounts for the effects of dynamic routing, congestion, and other naturally occurring noise. This propagation model enables *VerLoc* to convert measured times into geographical distances while accounting for the effects of noise on the confidence intervals.

We first assess *VerLoc*'s performance baseline in the absence of active adversaries, i. e., considering that all network nodes honestly report their location and measurements. We conduct extensive simulations to evaluate the performance of *VerLoc* under different conditions and understand the effects of parameters that affect the accuracy of the results. We present results for a challenging deployment scenario and show that even in sub-optimal conditions *VerLoc* is able to localize nodes within a median error range of 103 km and verify with accuracy 92 % the country where a node is located. Repeating the experiments in the wild provides even better results, with a median localization error of 60 km. We then evaluate *VerLoc* against an adversary that controls a subset of nodes, considering that malicious nodes may lie about their location, report fake timing measurements, and even manipulate pairwise measurements by delaying responses. We introduce a *confidence score* that qualifies location verification decisions and show that the score is effective for accurately distinguishing between true (honest) and false (adversarial) reported locations. We find that adversaries need to control more than 20 % of nodes to begin to degrade the location verification accuracy for honest nodes, and more than 30 % to trick *VerLoc* into accepting fake locations.

2 Preliminaries

2.1 Problem Statement

We consider a network of servers, which we refer to as *nodes*, that are geographically disperse and work together to enable a service, e. g., the relays that constitute the Tor network or the peers that are part of the Bitcoin network. Note that the network may provide services to *end clients* that do not publicize their contact and location information or take part in the verification process. We consider that only the nodes that form the network infrastructure participate in the *VerLoc* protocols.

Network nodes announce their geographical location and conduct a limited number of pairwise RTT measurements that they also broadcast. Based on the set of claimed locations and pairwise measurements, *VerLoc* allows everyone to verify all the claimed locations in a fully decentralized manner. *VerLoc* is designed to function in an adversarial environment where a subset of malicious nodes coordinate to claim false locations. The goal of *VerLoc* is to distinguish between true and false locations with high accuracy, even in the face of random network noise and active adversaries that control a subset of colluding nodes. Considering an adversary that claims false node locations and manipulates reported measurements, *VerLoc*'s main security goals can be formulated as follows: first, an adversarial node n_a at geo-location l_a cannot successfully claim being at a distant location l'_a (e.g., that is in a different country); and second, honest nodes that correctly follow the protocols and report truthful information are successful in verifying their geo-location.

Network model. We model the network as a set of N nodes n_i with $i = 1..N$. Each node n_i broadcasts a descriptor with its public key pk_i, its network address IP_i, and its geographical location loc_i (ideally with an error of less than 10 km), specified by latitude and longitude. This information is periodically broadcast and nodes are 'committed' to their keys, address and location until the next update. We also assume that it is possible for any two nodes n_i and n_j to communicate directly via the Internet, i. e., the network graph is fully connected.

Broadcast channel. We assume that each node knows the full set of $N-1$ other nodes, with their *node descriptors* (including pk_i, IP_i, and loc_i). This may be achieved in different ways. In anonymous communication networks this informa-

tion is typically updated in a consensus document published every hour or few hours, while in peer-to-peer networks maintaining blockchains the information may be continuously updated via gossip protocols. The timing measurements collected by nodes, which are needed to verify locations, must also be broadcast to ensure public verifiability of results. *VerLoc* thus requires a broadcast channel. We propose using a blockchain that acts as a public, append-only log maintained in a decentralized manner by the nodes. Note however that this blockchain can be replaced by any technology that provides secure broadcast with Byzantine fault-tolerance [32]. The key security features required by *VerLoc* from the channel are integrity and availability, i.e., that once uploaded, information is publicly available to all participants and cannot be altered. Node descriptors must be digitally signed to ensure that they have been generated by the node associated to the descriptor's public key and have not been altered by others. In turn, the list of descriptors of the nodes that constitute the network for a period of time is jointly signed by the entities maintaining the broadcast channel. We assume all participants can authenticate the channel without being tricked into believing that a separate channel (controlled by the adversary) is the authentic one containing *VerLoc*'s information. Furthermore we assume that it is not possible for the adversary to censor participant's read or write access to the channel, i.e., all nodes are able to broadcast their descriptor and measurements, and to read the descriptors and measurements of all the other nodes.

Epochs. We consider that nodes commit to being available at a location for a finite amount of time. The network is updated periodically, e. g. every few hours, with the *epoch* length being dependent on the expected churn in the network. Before the start of an *epoch*, nodes broadcast their updated keys, addresses and locations. The *VerLoc* protocols run during the epoch and produce results that enable identifying malicious nodes and possibly excluding them from the next epoch.

Timing information. *VerLoc* relies on *timing information* to estimate locations, using the relation between measurable transmission times and geographical distances between nodes, which is bound by the speed of light. More precisely, the transmission time defines the area that can be reached within that amount of time – with larger distances being impossible to reach, as that would imply transmissions speeds that are faster than light.

Timing probes. Nodes in the network probe a subset of other nodes and measure the round trip time (RTT), i. e., the time elapsed between sending a request and receiving a response. These pairwise measurements do not require any central authority and can be conducted using existing protocols such as the Internet Control Message Protocol (ICMP).

Trilateration. Combining transmission times measured from reference points situated in different directions allows to narrow down the location of a node. As all nodes measure various other nodes in *VerLoc*, there is redundancy in the overall set of network measurements. This redundancy allows to detect inconsistencies created by the malicious activity of adversarial nodes, as well as distortions introduced by exceptionally bad network conditions.

Inference of geographical coordinates. Based on all the claimed node locations and reported timing measurements, *VerLoc* estimates the most likely geographical coordinates (latitude, longitude) of each node and checks the distance to the node's claimed location.

Most likely geographical zone. We consider that space may be divided into countries, regions, zones, or any other territorial division, with each location (and therefore each node) belonging to precisely one zone. In addition to the most likely geographical coordinates, *VerLoc* computes the probability that a node is located within a zone.

Confidence scores. Finally, *VerLoc* compares the set of empirically measured propagation times with the times one would expect given the locations claimed by all nodes and the propagation model. This is used to define a *confidence score* that expresses the discrepancy between expected and measured times. A low confidence score is indicative that a node localization result (coordinates as well as zone) may be wrong and possibly malicious.

2.2 Information Propagation Model

Considering propagation *speed*, the transmission *time* between two nodes is proportional to the *distance* between them. However, network effects introduce noise and variance in the transmission speed, and consequently error when estimating nodes' locations from measured times. To provide real-world capabilities, *VerLoc* must be robust to realistic levels of noise and account for the actual speed function in the underlying Internet. We conduct experiments where we measure the timings of Internet transmissions between servers placed in different locations around the world. From these experiments we distill a realistic propagation model that is shown in Figure 1. The steps we took to derive this model are explained in detail in Appendix A. Note that the propagation model is pre-computed *once* and provided as a component of *VerLoc*. While a better model can be created with a larger number of measurements, once it is obtained it does not need to be updated until the Internet infrastructure undergoes a significant enough update to change its overall propagation characteristics.

We find that transmission speed is, in practice, more complex than a simple constant due to *background noise* caused by varying transmission medium characteristics, asymmetric and dynamic routing, congestion, and a host of other effects. As shown in Figure 1, propagation speed and background noise depend on end-to-end distance. By applying a fitting function, we can model the scattered and noisy transmission speeds as an estimate $f(x)$ that provides us the most likely propagation speed for a given distance x. The speed function $f(x)$ (and its inverse $f^{-1}(t)$, which converts times to distances) is used

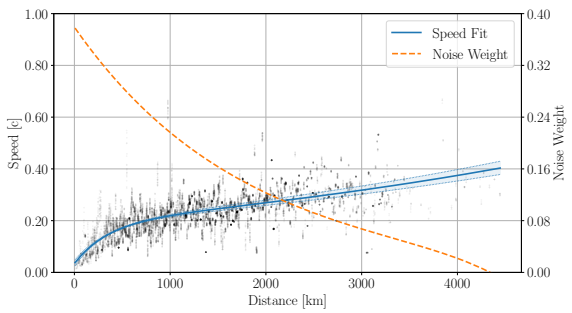

Figure 1: Propagation Speed Model. The scattered points show the individual speeds of ICMP traffic RTTs; the *blue* is the speed fit; the *orange* line summarizes the impact of background noise.

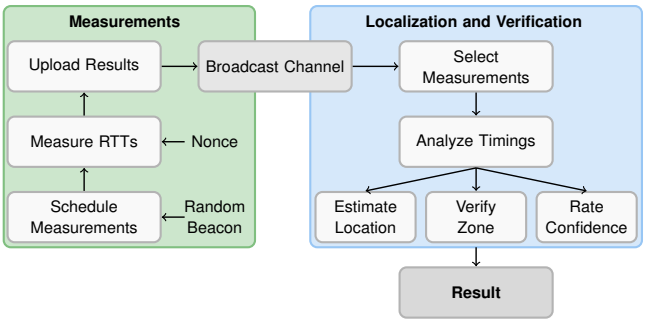

Figure 2: Complete measurement and analysis process. The green part includes steps involved in conducting RTT measurements, the blue part depicts the location estimation.

to estimate node locations (§3.3.1) and compute confidence scores (§3.3.3). Furthermore, the variance of observed times for similar distances allows us to capture background noise characteristics. We find that noise is lower for longer transmissions. To account for this effect, we compute a noise weight that we later use for localization.

Our propagation model is based on *real-world data* and therefore captures the actual transmission characteristics of the Internet. We base our simulation experiments on this model, and thus our sampled transmission times incorporate congestion latency and variance due to dynamic routing that is characteristic of the Internet.

3 System Concept

In this section we introduce the architecture and core system components of *VerLoc*. We explain how the *measurement component* schedules the random selection of references and the symmetric timing measurements, and how the *localization and verification component* analyzes the measured timings to produce location verification results.

3.1 System Components

As illustrated in Fig. 2, *VerLoc* consists of a **measurement component** (green) that outputs timing measurements for selected pairs of nodes, and a **location verification component** (blue) that analyzes those timings to estimate the locations of network nodes. In addition, the nodes collaboratively maintain a publicly accessible **broadcast channel** [32], allowing all nodes to broadcast their information and access the information broadcast by others. This makes the verification process fully *decentralized*, as everyone can locally compute localization results based on broadcast data. The blockchain stores the public network parameters, the node descriptors (public key, IP address, claimed location), a per-epoch random beacon, and the reported RTT measurements.

3.2 Measurement component

The measurement component involves three tasks that are executed by all network nodes:

1. **Schedule Measurements.** In each epoch, the network derives nodes' reference sets, which determine the schedule of pairwise measurements, using as seed a random beacon [46] (§3.2.1).

2. **Perform Measurements.** Nodes conduct pairwise measurements according to the scheduled reference sets by sending timing probes and recording the observed RTT (§3.2.2).

3. **Upload Measurements.** Nodes broadcast the minimum measured RTT for each node in their reference set (§3.2.3).

3.2.1 Schedule Measurements

VerLoc is designed to function in a fully decentralized fashion, without relying on any trusted authorities or third parties, and with all nodes performing the same tasks. To provide robustness, it is crucial that the reference sets are chosen in a way that cannot be biased by an adversary. Otherwise, the adversary may manipulate and exploit reference sets, e. g., selecting adversarial reference sets to successfully verify false locations and reject true locations. To prevent this, *VerLoc* assigns reference sets pseudorandomly.

VerLoc's reference set construction algorithm scales to arbitrarily large networks while maintaining a *constant* (rather than quadratic) complexity in terms of the number of measurements conducted per node. As shown in Sect. 4.3, the localization accuracy of *VerLoc* increases with the number of references per node, but the improvement has diminishing returns and, after a certain point, additional references consume resources without significantly improving performance.

We empirically determine that the best tradeoff is between 40 and 80 references per node, and use those values in our experiments.

Reference Set Construction. Reference sets are derived from the node's public keys pk_i and a random beacon x that is jointly computed by nodes and published in the blockchain once all public keys have been committed. Alternatively, the random beacon may be obtained from an external source of randomness [29], e.g., the hash of the first bitcoin block published after the start of the epoch. The beacon must only become available once all nodes have committed to their public keys and are ready for a new run of the *VerLoc* protocol. The key security requirements are that the same x is available to all nodes, and that the adversary can neither determine the value of x, nor predict it before committing to its node public keys. Given x, everyone can derive a random hash h_i for node i as: $h_i = H(x||pk_i)$, where $H()$ is a hash function [5].

We denote as R_i the *reference set* of node n_i. Given h_i, nodes follow Algorithm 1 to derive an initial set of t references to be included in R_i. These t references are only a part of a node's reference set R_i. In *VerLoc*, references are *symmetric*, meaning that $n_j \in R_i \iff n_i \in R_j$. Note that n_j can verify that n_i correctly selected it, using h_i and Algorithm 1. Nodes complete their reference set R_i following Algorithm 2.

Algorithm 1 Derive initial reference set R_i for node n_i

```
R_i := ∅
y := h_i
while |R_i| < t do
    r := y mod N;
    if r ∉ R_i and r ≠ n_i then
        R_i.add(r)
    end if
    y := Hash(y)
end while
return R_i
```

Algorithm 2 Complete reference set R_i

```
for j = 1..N do
    if n_j ∉ R_i and n_i ∈ R_j then
        R_i.add(n_j)
    end if
end for
return R_i
```

With this algorithm, the complete reference set R_i will have a variable size depending on the instance, as it is the sum of two components: the t references (constant number) derived from h_i, and an additional t' references (variable number) that are derived from all the other $h_j, i \neq j$. Note that the t nodes that are *already* included in R_i do not add any new reference to R_i even if $n_i \in R_j$. Given the network size N and the number of references t, a new node $n_j \notin R_i$ is added to R_i with probability $\frac{t}{N}$, and this applies to all the $N-t$ nodes that are not in the initial R_i. The probability of t' taking a certain value k thus follows a binomial distribution:

$$Pr\left[t' = k; N-t, \frac{t}{N}\right] = \binom{N-t}{k} \cdot \left(\frac{t}{N}\right)^k \cdot \left(1-\frac{t}{N}\right)^{N-t-k} \quad (1)$$

We select the parameter t to ensure that with overwhelming probability all N nodes have sufficient $t+t'$ references.

Symmetric Measurements. We use symmetric measurements for two reasons. First, they help leveling out noise and effects of asymmetric routing, as the timing $RTT(n_i \to n_j)$ might differ from $RTT(n_j \to n_i)$. Moreover, burst noise in one direction does not necessarily occur in the other direction and thus averaging both directions improves the overall robustness to noise. Second, averaging the times measured in both directions improves the robustness of *VerLoc* not just towards random noise, but also active attacks. In settings where of two nodes involved in the measurement one is honest and the other malicious, the adversary could try to report a very short transmission time to manipulate (*speed up*) the average transmission time. Similarly, the average could be *slowed down* to a desired number by the adversary reporting a very large time. *However, these attempts at manipulating the average are easily detectable by the confidence score (§3.3.3).* Speeding up the average transmission means that the adversary must contribute a measurement that is significantly *too fast*. Overly fast transmissions (faster than $2/3 \cdot c$) violate the upper speed bound and decrease the confidence score. Reporting a measurement that is significantly *too slow* will violate the lower speed bound and also result in marking the measurement as unreliable. This lowers the confidence score for both the target and the adversary nodes, meaning that adversaries that lower the confidence score of honest nodes will cause their *own* confidence score to diminish in equal measure.

3.2.2 Conduct Measurements

Given a set of references R_i for node n_i, *VerLoc* conducts pairwise measurements $RTT(n_i \leftarrow r_j)$ and $RTT(n_i \to r_j)$ between n_i and all its reference nodes $r_j \in R_i$. A node n_i conducts a measurement $RTT(n_i \to r_j)$ by sending timing probes to r_j, to which r_j responds as fast as possible, and recording the round trip time (RTT) of the responses. In their simplest form, timing probes can be implemented, e. g., as ICMP echo requests.

Freshness. To guarantee freshness, the node that initiates the timing probe includes a locally generated random nonce. When using ICMP, it is possible to encode the nonce in the variable-length data field. The responding node has to copy this nonce in the response. This prevents adversaries from *speeding up* measurements. More precisely, it is always possible for a node to hold back the response to incoming probes,

increasing the RTT and faking a longer transmission distance (§5.1). By including an unpredictable nonce, an adversarial node cannot respond to an incoming probe *before* it arrives, which eliminates the capability to fake a shorter distance. The nonce can be hashed with a previously established shared secret to protect against man-in-the-middle adversaries (§5.6).

Multiple Probes and Minimum RTT. The number of timing probes sent between a pair of nodes to conduct one measurement strikes a tradeoff between measurement accuracy and overhead. Increasing the number of probes allows to better overcome high-frequency, high-delta noise at the cost of sending more messages and taking longer to complete the protocol. As with measurements taken to infer the propagation model, nodes take the *minimum* RTT of all the timing probes exchanged with another node as the *least noisy* value. In our experiments we use series of 200 probes and extract the minimum value for each $RTT(n_i \rightarrow n_j)$.

3.2.3 Upload Measurements

Nodes broadcast the minimum measured RTT with each of their references. These measurements are the input for the localization and verification steps.

3.3 Localization and Verification

The analysis steps of *VerLoc* are executed by all nodes locally using the reported RTT measurements as input. *VerLoc* outputs three types of results for each node n_i: (1) an estimate of its location coordinates $\widehat{loc_i}$, (2) a binary verification decision for the node's claimed localization zone z_i, and (3) a score c_i that indicates *VerLoc*'s confidence in the previous two results.

3.3.1 Estimate Location Coordinates

VerLoc uses a gradient descent algorithm to estimate the geographical location of n_i. This optimization model is iterative and repeats three steps (illustrated in Figure 3) until it finds an estimated location $\widehat{loc_i}$ with minimum error. The process is based on the principle of trilateration [24, 27, 49] and defined as follows.

Define Candidate Location. In the first step, we compute the pairwise great circle distances *dist* between the claimed locations of nodes in the reference set $r_j \in R_i$ and a *possible* candidate location $\widehat{loc_i}$ for node n_i. Note that we assume that most nodes are non-adversarial and claim a *correct* location. The initial $\widehat{loc_i}$ is an educated guess for the location of n_i that is adjusted throughout the optimization steps to find the best result. The resulting vector $\overrightarrow{dist}$, of size $R = |R_i|$, contains the distances between all references r_j and candidate $\widehat{loc_i}$.

Estimate Distances. We average measurements in both directions to estimate the distance between n_i and r_j (§3.2.1).

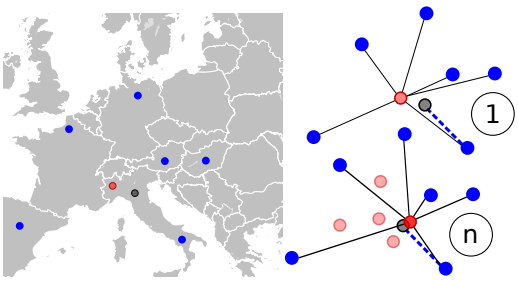

Figure 3: Node localization process (§3.3.1). At the start of the localization process, we collect claimed locations and timing measurements from each of the references (blue circles) of a target node (black circle). We guess an initial location (red circle) for the target and then apply the iterative optimization. In the first step ①, we take the distance between each reference's claimed location and the guessed location (black lines). We compare this with the distance that corresponds to the *measured* RTT (blue dashed line) and evaluate the discrepancy between both values. In the following steps ⓝ, we use a gradient descent to adjust the guessed location (red circle) until we find the solution with the least discrepancy.

$$RTT(n_i \leftrightarrow r_j) = \frac{RTT(n_i \rightarrow r_j) + RTT(n_i \leftarrow r_j)}{2} \quad (2)$$

$RTT(n_i \rightarrow r_j)$ and $RTT(n_i \leftarrow r_j)$ are the *minimum* RTTs measured in each direction. As described in Section 2.2, we can translate transmission time into distance by applying the inverse speed function $f^{-1}(t)$. This results in a second distance vector $\overrightarrow{dist_{RTT}}$ that contains $f^{-1}(RTT(n_i \leftrightarrow r_j))$ for $r_j \in R_i$

Apply Error Function. In the third step, we compare the candidate distances $\overrightarrow{dist}$ with the distances $\overrightarrow{dist_{RTT}}$ derived from the measured RTTs. The delta between both vectors $\Delta(\overrightarrow{dist}, \overrightarrow{dist_{RTT}})$ expresses the discrepancy between the reported RTTs and the candidate location $\widehat{loc_i}$. We apply the root-mean-square error (RMSE) as an error function to evaluate how consistent $\widehat{loc_i}$ is with the reported measurements. We use an iterative optimization to minimize the RMSE for possible values of $\widehat{loc_i}$:

$$\underset{\widehat{loc_i}}{\arg\min} \sqrt{\frac{\sum_{j=1}^{R} (\Delta_j \cdot \omega_j)^2}{R}} \quad (3)$$

$\Delta_j = |dist(\widehat{loc_i}, r_j) - f^{-1}(RTT(n_i \leftrightarrow r_j))|$ is the difference between the candidate and the measured distances for reference r_j, R is the number of references in R_i, and ω_j is a distance-dependent *weighting* factor that accounts for noise effects. We derive ω_j empirically as part of the propagation model introduced in Section 2.2.

End Result. At the end of this process, we obtain a location estimate $\widehat{loc_i}$ that best fits the reported RTT measurements between n_i and its references.

3.3.2 Verify Zone

In localization problems, the specific geographic coordinates are often part of an area that has a semantic significance, e. g., a country, jurisdiction, or zone under the control of a given actor. In this case, all the points within a zone are considered equivalent. In addition to the coordinates that best approximate the node's location, *VerLoc* can ascertain whether the node is located within the zone z_i that contains the claimed location loc_i. The process illustrated in Figure 4 consists of the following steps.

Derive Target Area. In a first step, *VerLoc* translates the measured RTTs into a vector of distances. Transmission speeds on the Internet range from 0.22 c to 0.67 c [25, 34]. In order to find the *largest* target area that could possibly meet the constraints of all measurements, we use an upper bound transmission speed of $2/3 \cdot c$ to compute the distance $dist_{max}(n_i \leftrightarrow r_j) = 2/3 \cdot c \cdot RTT(n_i \leftrightarrow r_j)$.

We sort the distances in ascending order and pick the first element in this list, i. e., the reference appearing to be closest to the target node n_i. We "draw a circle" around this first reference of radius $dist_{max}(n_i \leftrightarrow r_j)$. The circle describes the area that could have been reached in the measured time. This circle is an initial area where n_i must be located, which is further narrowed down with each additional reference.

Shrink Target Area. We then proceed iteratively with the next references of the sorted list. For each reference r_j, we draw a new circle around r_j of radius $dist_{max}(n_i \leftrightarrow r_j)$ and compute the *intersection* with the previous circles. In this iterative process, we narrow down the target area step by step and exit the process when new references do not shrink the target area any further. The approach is robust to network distortions, as occurrences of high background noise lead to longer distances $dist_{max}(n_i \leftrightarrow r_j)$, whose resulting intersections are not overly restrictive.

Apply Grid. In the final step, we apply a grid to the target area resulting from all intersections. We compute a likelihood score for each point in the grid based on Δ_j, which expresses how consistent that point's location is with the measured RTTs. We normalize the scores to obtain a probability distribution, and then sum the scores of the points within each zone that overlaps with the target area. The zone that accumulates the highest mass is then compared to the claimed zone for the verification decision. In the example shown in Figure 5 *VerLoc* would output a positive zone verification if the node has claimed to be in Italy, and negative otherwise. Note that it is also possible for *VerLoc* to output the probability score of each zone instead of just the zone with the maximum score.

3.3.3 Confidence Scores

Confidence scores express the degree to which measured RTTs are consistent with the claimed locations of all nodes given the propagation model. This allows to reject decisions where the evidence is inconclusive regarding the node's location. In the following, we introduce the *generic* concept of confidence scores, whose parameters we later adjust to detect adversarial timing manipulations (§5).

Speed Bounds. We consider bounds b_l and b_u that define the minimum and maximum propagation speeds for a transmission. The upper bound $b_u = 2/3 \cdot c$ serves as a sanity check and describes the maximum transmission speed that we usually observe on the Internet. The lower bound b_l is the lower 95 % confidence bound of the speed fit (§2.2). A tolerance factor τ accounts for transmission noise that slows down packets beyond the considered speed limit:

$$b_l = l(x)(1-\tau) \qquad 0 \leq \tau \leq 1 \tag{4}$$

Apply Confidence Scores. Consider a target node n_i with a set of references $r_j \in R_i$ and reported timings $RTT(n_i \rightarrow r_j)$ and $RTT(n_i \leftarrow r_j)$ for each direction. The node's confidence score c_i represents the percentage of reference measurement pairs within bounds, with the highest possible score being 1 and the lowest 0. More precisely, we count a 1 for every pair of measurements that stays within the bounds and a 0 for every pair where at least one direction ($RTT(n_i \rightarrow r_j)$ or $RTT(n_i \leftarrow r_j)$) violates at least one bound (b_l or b_u). We then normalize the count dividing by the number of references $|R_i|$.

A threshold can then be used to either accept or reject n_i's localization results based on its confidence score c_i. In Section 5 we show how confidence scores can be used as countermeasure to distinguish benign from manipulated measurements.

4 Performance Baseline

We begin our evaluation of *VerLoc* with a study of its performance baseline in a non-adversarial simulation setup.

4.1 Experimental Setup

We evaluate *VerLoc* with simulations that account for the propagation characteristics of real-world networks. The procedure includes a preparation phase, where we randomly generate a network, and a simulation phase, where we apply *VerLoc*.

Preparation. We generate a network of N nodes n_i, each of which has a randomly chosen true location loc_i in a zone z_i among fifteen of the most populated European countries: Germany, France, United Kingdom, Italy, Spain, Ukraine, Poland, Romania, Netherlands, Belgium, Czech Republic, Hungary, Austria, Switzerland, and Slovakia. Based on these randomly generated locations we generate a propagation matrix that contains pairwise timing measurements for all possible pairs

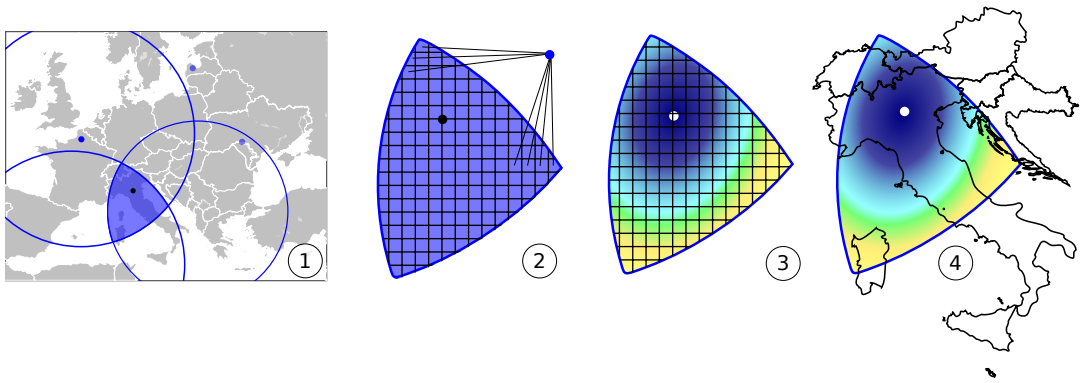

Figure 4: Zone Verification Process (§3.3.2). In the first step ①, we generate an intersection area containing all possible locations for the node. In the second step ②, we apply a grid to the intersection and compute the distances between each point in the grid (squares) and each reference node (blue point). In the third step ③, we assign a weight to each point in the grid that describes the probability of reaching the point from the reference in the measured time, darker colors represent a higher weight. In the final step ④, we sum the weights of the grid points within each country, and pick the country with the largest sum (Italy, in this example).

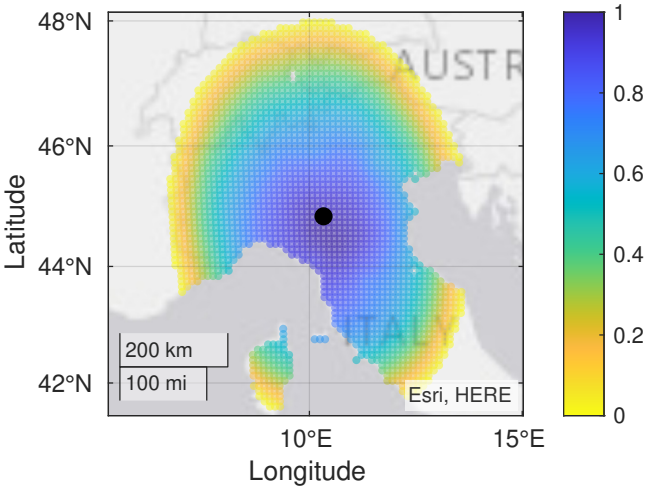

Figure 5: Sample Target Grid. The black point is the location of n_i (ground truth); the colored area shows the weighted grid; darker colors indicate a higher likelihood score for a point in the grid. For the verification decision, we pick the country with the largest sum of likelihood scores.

of nodes in the network. We sample the transmission speeds, times, and noise from our empirical propagation model (§2.2).

Simulation. We first generate the set of references $r_j \in R_i$ for each node n_i. For this, we pick uniformly at random t references out of the available $N-1$ nodes and extend the set to ensure that all measurements are symmetric (cf. Alg. 2). For each measurement pair we look up timings from the pre-computed propagation matrix. We then apply the method described in Section 3.3.1 to estimate n_i's location $\widehat{loc_i}$ and the method of Section 3.3.2 to verify its zone, as defined by country borders.

4.2 Metrics

We use two metrics to evaluate the performance baseline. First, we measure the *location error* of a node n_i as the great circle distance between the estimated and actual node locations, $dist(loc_i, \widehat{loc_i})$. Second, we compute the *zone verification rate* as the fraction of nodes for which the highest weight zone matches the ground truth of the node's location zone.

4.3 Experiments

Number of References. As initial step to adjust *VerLoc*'s parameters, we analyze how the number of references influences node localization accuracy and zone verification rates. The number of references $|R_i|$ of node n_i strikes a tradeoff between the overhead and performance of *VerLoc*. References must be picked randomly to prevent attacks. Thus, we cannot optimize the choice of R_i to maximize proximity to n_i or diversity of directions, which would increase localization accuracy. Increasing the number of references is the next best option to ensure diversity of measurement directions and to level out noise.

To find a suitable target number R of references, we compute the average location error and zone verification rate for different values of R, and show the results in Figure 6. We observe a significant performance improvement in the range of $R = 10$ to $R = 80$ references, which then offers diminishing returns for larger values of R. For our simulations we choose $R = 80$ as lower bound for the number of references, and set parameter $t = 50$ so that reference sets are larger than $R = 80$ in at least 98 % of cases, i. e., $|R_i| \geq R$ for most nodes (cf. Eq. 1). In the real-world experiments presented later (§6) we test both $R = 40$ and $R = 80$ and find that the difference in localization accuracy is less than 0.5 km.

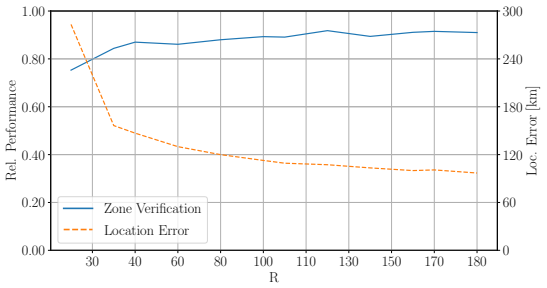

Figure 6: Performance for Localization and Verification. The zone verification rate (left y axis) shows the fraction of correct zone verifications. The localization error (right y axis) shows the distance between the estimated and true node locations.

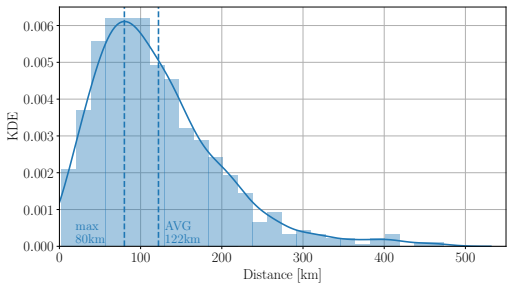

Figure 7: KDE Localization Error: Distribution of distances between the estimated and the true location of nodes.

Table 1: Simulation Parameter Setup.

Parameter	Notation	Value	Section
Nr Network Nodes	N	1000	
Network Node	n_i	–	
Keys	(pk_i, sk_i)	–	
True Physical Loc.	loc_i	–	§4.1
Estimated Loc.	$\widehat{loc_i}$	–	
Claimed Loc.	loc_i^A	–	
True Physical Zone	z_i	–	
Estimated Zone	$\widehat{z_i}$	–	
Base References	t	50	§4.3
Reference Set	$r_j \in R_i$	$\lvert R_i \rvert \geq 80$	
Nr Adversarial Nodes	$\lvert A \rvert$	50 .. 300	§5.2
Adversarial Nodes	$a_j \in A$	–	
Claimed Nodes	$c_k \in C$	$C \subseteq A$	
Confidence Threshold	υ	0.2	§5.4
Tolerance Factor	τ	0.01	

Baseline Parameter Values. We document the *VerLoc* parameters in Table 1. The first two blocks describe the network setup and the number of references we determined in the performance baseline. The following two blocks are dedicated to the performance of *VerLoc* in an adversarial setting, which we discuss in the following section.

Localization and Zone Verification Results. In our experiments the median localization error is 103 km and the average is 122 km. The distance between true and estimated locations is distributed as shown in Figure 7. To get a sense of how these distances compare to European country sizes, we empirically evaluate the likelihood that the error will move the node across a country border. As expected, shifts between small neighboring countries are more likely to happen, but *VerLoc* still achieves on average 92 % accuracy when verifying the country in Europe where a node is located. We note that experiments in the wild outperform simulation results (§6.2), confirming that our simulations represent a particularly difficult deployment scenario.

5 Security Analysis

We consider an adversary that participates in the network with several malicious nodes. The adversary claims false locations (different from the nodes' true physical locations) for a subset of those nodes and manipulates measurements consistently with the fake claimed locations. Claiming false locations undermines the localization and verification capabilities of *VerLoc* compared to the baseline. We extend here the initial network model (§2.1) to account for adversarial setups.

5.1 Adversarial Timing Manipulations

We consider a network of N nodes, of which a subset A of nodes is adversarial, $A \subset N$. The adversary claims false locations $loc_k^A \neq loc_k$ for a subset C of adversarial nodes, $C \subseteq A$. The adversary *controls* all nodes in A and can thus manipulate timings whenever an adversarial node participates in a measurement.

Given a malicious target node $c_k \in C$ and its reference set R_k, there are three possible scenarios for the target and reference pairings $c_k \leftrightarrow r_j$.

Perfect Manipulation. A *perfect manipulation* is possible when both the target node c_k *and* the reference $r_j \in R_k$ are under adversarial control, i. e., $c_k \in C$ *and* $r_j \in A$. In this case the adversary contributes spoofed times $RTT(c_k \rightarrow r_j)$ and $RTT(c_k \leftarrow r_j)$ for both directions. The spoofed timings match the expected propagation time for the claimed locations:

$$RTT(c_k \rightarrow r_j) = RTT(c_k \leftarrow r_j) = \frac{dist(loc_k^A, loc_j^A)}{f(dist(loc_k^A, loc_j^A))} \quad (5)$$

More precisely, the adversary first computes the distance between the *claimed* location loc_k^A and the adversarial reference node loc_j^A. Note that $loc_j^A = loc_j$ if $r_j \in A$ but $r_j \notin C$,

while $loc_j^A \neq loc_j$ if $r_j \in C$. This distance serves as an input to the empirical speed function $f(x)$ (cf. § 2.2). We assume that all the parameters of *VerLoc* are known to the adversary, who can apply the propagation model to compute timings that match the claimed distance. The adversary can slightly alter reported RTT values with noise to avoid submitting suspiciously identical numbers.

The following two scenarios cover cases in which the reference node is *not* adversarial, i. e., $c_k \in C$ but $r_j \notin A$.

Slowing Down. If the claimed location loc_k^A of c_k is *further* away from reference r_j than the true location loc_k, i. e., if $dist(loc_k^A, loc_j) > dist(loc_k, loc_j)$, then it is possible for the adversary to *slow down* incoming timing probes $(c_k \leftarrow r_j)$. Slowing down means that the adversary delays its response to the incoming probe in order to bring the $RTT(c_k \leftarrow r_j)$ measured by r_j close to the value that would be observed if c_k was indeed placed in loc_k^A.

As in the previous case, we consider the adversary is able to tamper with timings *in both directions* to perfectly fit the claimed location. In one direction, the adversary contributes a spoofed timing, while in the other direction it adds latency to manipulate the timing reported by the (honest) reference node r_j.

No Manipulation. If the claimed location loc_k^A of c_k is *closer* to the benign reference r_j than its true location loc_k, i. e., if $dist(loc_k^A, loc_j) < dist(loc_k, loc_j)$, then the adversary *cannot* manipulate the measurements taken and reported by r_j, as he cannot speed up probes [27] or reply before receiving the reference's probe and seeing the included nonce.

5.2 Experimental Setup

In an adversarial setup, we are interested in the general system performance (accuracy of localization and zone verification rate) for honest nodes, as well as the number of successfully claimed false locations. To this end, we adjust the *network setup* and applied *metrics*.

Network Setup. As in the performance baseline experiments, we simulate networks of $N = 1000$ nodes placed in random locations across Europe and use a reference set size of $R = 80$.

We additionally define a subset A of adversarial nodes that can manipulate measurements and a subset $C \subseteq A$ of nodes for which the adversary claims false locations. We randomly pick fake claimed locations but ensure that the falsely claimed location is in a different country than the actual location of the adversarial node.

We first run a simulation considering true locations for all nodes. We then substitute the propagation times for all measurements in which the adversary can either apply a perfect manipulation or slow down incoming timing probes. We re-apply the localization and verification methods for all the nodes that were affected by the adversarial activities. This includes all nodes $c_k \in C$ with false claimed locations, but also all the benign nodes in their reference sets R_k. We do so to examine all the discrepancies that the adversary introduces when introducing bogus locations and measurements

Metrics. To measure the performance of *VerLoc* in an adversarial setup, we extend the initial performance metrics (§4.2) with a set of `true/false`, `positive/negative` results. In contrast to the benign setup, these results now include accept/reject decisions for individual nodes.

We denote a `positive` decision as an accept, i. e., a decision where the confidence score is sufficiently high; a `negative` is a rejected decision where the confidence falls below a defined threshold υ. Furthermore, we treat a result as `true` whenever it matches the ground truth and as `false` when it contradicts the ground truth:

- **TP** A true positive decision denotes an *accept* for a node that reports its true location $loc_i \in z_i$, meaning that *VerLoc* believes the node to be in the correct zone z_i.
- **TN** A true negative decision denotes a *reject* for an adversarial node claiming to be at a false location $loc_k^A \in z_k^A$ such that $z_k^A \neq z_k$.
- **FP** A false positive decision denotes an *accept* for a node with a false claimed location $loc_k^A \in z_k^A$, meaning that *VerLoc* believes the node to be in the falsely claimed zone z_k^A.
- **FN** A false negative decision denotes a *reject* for a node whose claimed location $loc_i \in z_i$ was actually true.

In the following, we use a *confidence score* to identify fake node locations. We then analyze the attack performance for an increasing number of adversarial nodes to test the *breaking point* of the system.

5.3 Attack Success

In a first step, we analyze the success of the adversary in claiming false locations for some of its nodes, considering a network of $N = 1000$, where each has at least $R = 80$ randomly chosen references. To this end, we allow the adversary to control $|A| = 50$ randomly chosen nodes and evaluate the attack success for an increasing number of false location claims $|C| = 5..50$. Note that in the case of $|C| = 50$ the adversary is claiming false locations for all adversarial nodes, i. e., $C = A$.

We show the number of successfully verified false locations in Figure 8, where we can see that in the absence of countermeasures the adversary is successful between a third and half of the times.

5.4 Confidence Scores

To harden *VerLoc* and mitigate the attack success, we use the confidence scores introduced previously (§3.3.3). We define a *reject threshold* υ to distinguish between benign and malicious location claims. We further define a *tolerance factor* τ

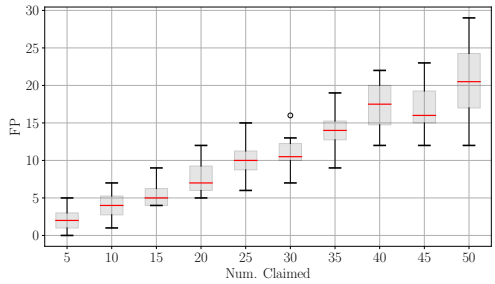

Figure 8: Adversarial success for a given number of false claimed locations (20 000 samples).

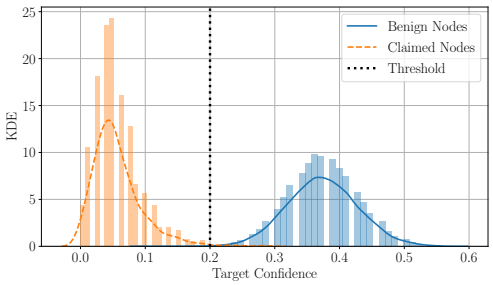

Figure 9: Distribution (kernel density estimate) of confidence scores. Benign (blue) and malicious (orange) nodes in a setup with $|C| = |A| = 50$ adversarial nodes claiming false locations. We choose a threshold $\upsilon = 0.2$ to distinguish both groups.

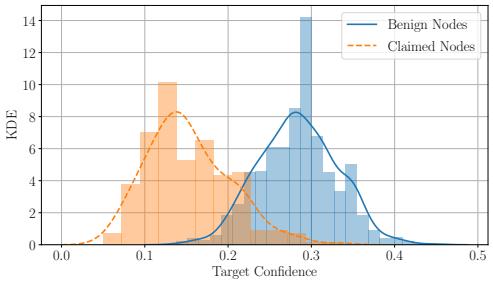

Figure 10: Distribution of confidence scores for a setup with 250 out of 1000 (25%) malicious nodes, all claiming false locations.

to account for background noise that disturbs the end-to-end timings. These are the two main parameters that determine accept and reject decisions based on the confidence scores.

The tolerance factor τ defines how much noise is tolerated, i. e., it relaxes the lower bound b_l on the speed, accepting even slower transmissions. Too many transmissions being too slow is precisely a distinguishing characteristic of false claimed locations. Whenever a false claimed location is *closer* to a reference r_j than the true location, the adversary is unable to manipulate the measured timing to make it shorter. Consequently, the RTT reported by the reference r_j will appear as a very slow (noisy) transmission.

The threshold υ defines the minimum decision confidence score required by *VerLoc* to accept a claimed location as verified. This parameter influences the tradeoff between false positive and false negative rates. An overly restrictive υ will reject many decisions, including those of benign nodes for which measurements are simply noisy. On the other hand, a very lax υ increases false positives, correctly verifying a larger number of honest node locations at the cost of also accepting some false locations as correct.

Tolerance Factor (τ). To evaluate how the tolerance factor τ influences the overall performance of *VerLoc*, we test values in the range of $\tau = 0.005..0.025$ considering a decision threshold $\upsilon = 0.2$. In our evaluation, we first study which configurations prevent the adversary from claiming *any* false location, i. e., cases in which $FP = 0$. We find that there is minimal variation for the accept and TP rates. More precisely, the acceptance rates are in the range of 95 % to 96 %, and the TP rates achieve 86 % to 87 % within the accepted decisions. For the simulation experiments we use a tolerance factor of $\tau = 0.01$.

Decision Threshold (υ). The decision threshold υ defines the minimum required confidence to accept a decision. Figure 9 shows the distribution of confidence scores for adversarial and benign nodes considering a scenario where the adversary controls 50 nodes, all of which claim false locations. As we can see in the figure, both groups can be perfectly distinguished with a decision threshold of $\upsilon = 0.2$, as there is no overlap between both distributions. We thus choose this value for υ.

5.5 Breaking Point of *VerLoc*

We have shown that *VerLoc* can reliably distinguish between manipulated and benign node decisions. This prevents adversarial success while maintaining a reliable verification and localization performance in situations where the adversary controls a limited percentage of nodes. However, we are also interested in identifying the *breaking point* of the system, i. e., the required amount of adversarial resources that degrades the performance of *VerLoc* to unacceptable levels. To this end, we gradually increase the fraction of adversarial control in the network (cf. Table 2). We observe:

- **Confidence Score Distribution.** An increasing number of adversarial nodes with false claimed locations introduces distortions in many measurements, and the manipulations lead to network-wide inconsistencies between claimed locations and reported timing measurements. Figure 10 shows how the confidence scores begin to overlap for benign and malicious nodes at 25 % compromise.

Table 2: System Breaking Point.

Claimed	Reject	TP	FP	FN	Recall
5 %	50	815	0	0	1.00
10 %	102	756	0	2	0.99
15 %	152	691	0	3	0.99
20 %	214	627	0	23	0.96
25 %	284	555	0	58	0.91
30 %	335	476	3	105	0.82
35 %	352	401	3	151	0.73

- **False Positives.** *VerLoc* begins to output FP results when reaching around a third of adversarial nodes with false location claims. While the adversarial success is limited in this setup ($FP = 3$), it indicates that at this point the adversary is capable of compromising the verification process to successfully claim false locations. Note that a FP only occurs when (1) the estimated zone $\hat{z}_i$ coincides with the zone that contains the location loc_i^A claimed by the adversary, and (2) the adversarial node has a confidence score larger than υ. A confidence score above υ does *not* lead to a FP if the estimated and claimed zones do not coincide.
- **False Negatives.** False negatives do not compromise the system's security directly but they have the risk of unfairly rejecting honest nodes from the system. Furthermore, an increasing FN rate indicates that *VerLoc* looses the ability to make reliable accept and reject decisions. We see that FN begins to increase when adversaries control 20 % or more of the network.

Conclusion. The confidence scores of *VerLoc* allow distinguishing true and false location claims. This protection mechanism is robust to adversaries that control up to about 20 % of nodes in the network while still providing high performance rates for all remaining honest nodes. Note that these results correspond to a network setup restricted to Europe, which is a challenging use case (§5.6).

5.6 Additional Threats

5.6.1 Framing Benign Nodes

Instead of trying to successfully claim false locations (FP) for malicious nodes, the adversary can attempt to frame benign nodes as claiming a false location (FN) by, e. g., contributing timings that violate bounds and reduce the confidence score of an honest node. An adversary that controls a fraction $\alpha = \frac{|A|}{N}$ of all the nodes is on average able to manipulate a fraction α of an honest node's reference measurements, in one of the two directions.[4] This is in contrast with claiming false locations, where the adversary controls *all* the reference measurements in at least one direction, and a fraction α in *both* directions.

An adversary that controls a fraction α of a node's references can *at most* lower the confidence score of the node by α, by making $\alpha \cdot R$ measurements that would otherwise be within bounds to be out of bounds. Considering the results shown in Figure 9 for benign nodes, more than half the references of a node fail the bounds test in the absence of attack, meaning that the adversary in practice will be only able to lower an honest node's score by less than α. Considering a threshold $\upsilon = 0.2$ for the confidence scores, an adversary has to control at least a fraction $\alpha = 0.2$ to start bringing down below υ the confidence score of a non-negligible fraction of benign nodes.

Note that, because measurements are symmetric, a bounds violation in a pairwise measurement affects both nodes involved, and thus the scores of the adversarial nodes themselves become lower as they attack more targets, meaning that the adversary has to trade scalability with detectability of the attack.

5.6.2 Strategic Locations for Adversarial Nodes

An adversary can improve the attack success by strategically positioning its nodes. For this strategic placement, two key characteristics can be exploited. First, zone verification errors where a node is believed to be in a different zone are more likely to occur in *small zones*, such that an error of less than a hundred kilometres is enough to shift the node to a different zone. Thus, an adversary placing the nodes in small zones can more plausibly claim that the observed discrepancies are due to natural network effects and noise rather than malicious manipulation. Within the localization constraints the adversary is subject to, placing the malicious nodes as close as possible to the border of the claimed zone increases the chances of adversarial success. Note that regardless of zone boundaries, it is always easier for an adversary to plausibly claim being at a nearby location rather than a faraway location.

Second, we investigate how the *directional diversity* of a reference set influences the accuracy of localization. In the best case, a node's references are situated in all directions to provide the highest possible measurement diversity. To illustrate why directional diversity is important, consider a case where multiple measurements have a lot of noise (extra latency). If the noisy measurements come from opposite directions, the discrepancies are leveled out and *VerLoc* arrives at a good estimation. In contrast, noisy measurements from a single direction push the estimated target node further away from its actual location. In our experiments, 95 % of failed verification for nodes in Spain, which is located in the SW corner of Europe, are associated with a unidirectional distribution of references, whereas this issue appears related to just

[4] The exact number of corrupt references in a node's set is given by a hypergeometric distribution with population size N, R draws, and $|A|$ special (malicious) objects. The variance of such distribution is low for a large R, making it unlikely that the adversary controls much more than a fraction α of references in a randomly chosen subset.

20 % of nodes in Romania, which has a more central location in the continent. Therefore, an adversary can more plausibly claim that a failed verification is due to natural causes when claiming locations with less directional diversity. Note also that the adversary has more influence on *VerLoc*'s results for a node when it is the only reference providing measurements for that node from a certain direction.

The study of adversarial node positioning strategies to achieve concrete objectives under specific constraints is left for future work. Such further research may, for example, evaluate strategies for successfully convincing the network that a number of adversarial nodes are located within a specific country (that, e.g., has a strong rule of law and favorable legislation, which increases trust in those adversarial nodes), when they are actually located somewhere else that is at a certain distance; or strategies for downgrading the confidence score of specific targets among the honest nodes.

5.6.3 Adversaries in the underlying network

Network adversaries placed in between two nodes may intercept probes and respond to them, causing nodes to measure RTTs that are impossibly small given the actual distance between them. Security can be strengthened towards such adversaries if the two nodes, who can authenticate each other using the public keys in the node descriptors, establish a shared secret s before exchanging probes, e. g., with an authenticated Diffie-Hellmann key exchange. Instead of simply copying the nonce in their response, nodes respond to probes with a hash of s concatenated with the nonce. A man-in-the-middle adversary who is physically between the two nodes cannot fake a shorter distance without access to s. Note that the time required to compute this hash adds to the measured RTT and thus needs to be factored in when building the propagation model. If the hash computation time is highly variable from one node to another, this adds noise that may decrease localization accuracy; if on the other hand the hash computation time is rather constant across servers, it can be easily accounted for in the propagation model without an impact on localization accuracy.

Alternatively, a network adversary can always slow down probes it intercepts. Note that this gives no additional advantage to an adversary that controls one of the nodes involved in the probe, who already has the ability to slow down the probe at will; but it does enable the adversary to delay probes sent between honest nodes. The effect of such an attack is to lower the confidence score of honest nodes, who now appear to be far away from many other nodes, or even all other nodes if the adversary fully controls the network connection of the victim. Note that distinguishing such delays from natural transmission delays due to poor network conditions is non-trivial, as both effects cause a lower confidence score. We argue that this is an acceptable effect, as the confidence scores not only represent adversarial manipulations, but also take into account bad network conditions. In both cases the score represents a lower confidence in the localization result.

5.6.4 Compromised Broadcast Channel

VerLoc relies on two types of node information to produce results: node descriptors (containing IP addresses, claimed locations and public keys) and recorded measurements. We assume that node descriptors are always securely broadcast even in the absence of *VerLoc*, as otherwise it is easy to completely disrupt the network and any functionality it could offer, beyond *VerLoc*'s node localization features. Note that in Tor this information is included in a consensus document signed by all directory authorities, while in Nym it is collectively signed by validators and published in the blockchain. The recorded measurements are however specific to *VerLoc* and not broadcast already as part of basic network orchestration. Therefore, we can expect *VerLoc* implementations such as the one described in the next section, where recorded measurements are made publicly available in ways that are more susceptible to adversarial attacks. An adversary could, for example, show different measurement results to different participants by serving a different result files depending on the IP address of the requester. In this case participants may arrive to different conclusions regarding the localization of some nodes. We note that public web pages with node information such as those maintained by Nodes Guru[5] make such attacks detectable, as participants may realize that their locally obtained *VerLoc* results do not coincide with results shared by others in public places. Thus, while such attacks are difficult to prevent in the absence of secure broadcast for measurements, they can be relatively easy to detect (and react to) given active community engagement, discussion and scrutiny of the system.

6 Experiments in the Wild

As final step we validate *VerLoc* with a real-world experiment where deployed nodes run a simplified prototype implementation.

6.1 Experimental setup

The simplified *VerLoc* prototype implementation was bundled with the Nym network's mixnode code version 0.10.1, released on May 25th. Thousands of nodes are actively participating in *VerLoc* measurements and publishing new results every 12 hours. More precisely, a list of mixnodes with IP addresses of all nodes involved in the network is publicly available.[6] Using these IP addresses, we can load the measurement results from a specific port directly at the node

[5] https://nodes.guru/nym/nymworld

[6] https://testnet-finney-explorer.nymtech.net/data/mixnodes.json

`http://[IP]:8000/verloc`. We have made publicly available the prototype implementation[7] used to obtain the results shown in this paper. The prototype includes the main implementation of *VerLoc* and a fetch and parsing script to obtain the node measurements. Our results are therefore fully reproducible and the prototype is available to conduct further studies.

The implementation functions as follows. As part of Nym's normal features (independently of *VerLoc*), mix nodes periodically download the full list of active nodes in the network, with their public keys and IP addresses. Nodes self-report location as part of their information, typically at the level of nearest town. In Appendix B we provide details on the quality of self-reported locations. A node that runs the *VerLoc*-enabled version of the software sends 200 ICMP echo requests to each of the other nodes in the network and records the measured RTT values. The minimum RTT per reference is made available at a defined port accessible through the node's IP address. We crawl and parse these files to retrieve the measurements. We then locally run our Python scripts on that data to compute localization results. The main differences between the prototype version that is currently deployed and the full proposed *VerLoc* system are:

- The prototype does not publish the measured RTTs in a blockchain but instead makes them publicly available as json files.
- In the prototype nodes measure *all* (thousands) other nodes in the network rather than a subset of 40 to 80 references. This is to allow us to collect a larger dataset and perform a more thorough evaluation. A complete *VerLoc* implementation conducts fewer measurements (two orders of magnitude less) and thus generates *much* less data so that it is feasible to publish RTT measurements in a blockchain.

6.2 Results

At the time of submission (June 2021) there are 7469 individual nodes in the Nym network. After crawling, parsing, and filtering for nodes running the *VerLoc*-enabled version 0.10.1, we were left with 3460 nodes for our evaluation. The main 'loss' of nodes running the right software version is caused by problems parsing or mapping the self-reported locations. In the current prototype implementation, node operators provide location as a string that should be the name of a city. We then look up the name in a publicly available list of 26569 world cities. We use the coordinates of the center point of the city as an approximation of the node's self-reported location. Unfortunately, the free text leads to parsing errors due to spelling mistakes and formatting issues. Furthermore, if nodes provide the name of a small place not in the list of world cities, our current prototype cannot assign a reported location to the node. In a more advanced version of *VerLoc*, the self-reported location should be standardized, e. g., by providing numerical fields for reporting latitude and longitude instead of a free text field for the location name—or substituted altogether by GeoIP locations.

[7] https://github.com/katharinakohls/VerLoc

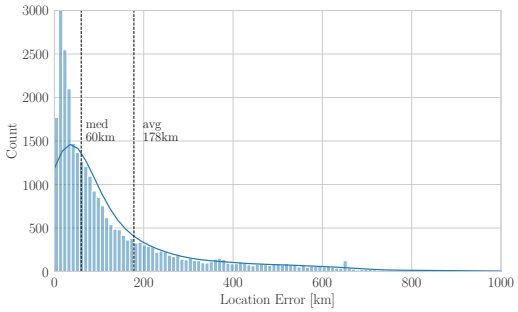

Figure 11: KDE Localization Error: Distribution of distances between the estimated and the true location of nodes.

For comparison to the simulation results (§4) we select nodes in Europe that have a sufficient number of measurements, which results in a subset of 943 out of the 3460 nodes. The filtered dataset with 943 nodes contains 852870 individual measurements, with an average of 902 references per node. For each selected node, we randomly choose R of its references, discarding and re-sampling if we get a duplicate location already in the reference set. We then retrieve the corresponding measurements from the dataset.

We tested $R = 80$ and $R = 40$ and the median localization error differed by less than 0.5 km. The results show a median localization error of 60 km and an average of 178 km, which are better results than those obtained in the simulations (§4). The distribution of error is shown in Figure 11. Compared to Figure 7, it is more skewed, with a larger number of rather accurate localizations and a long tail of error. This performance improves when applying confidence scores (§5.4). Although we do not expect an active manipulation of measurements in the collected data set, self-reported locations may be grossly inaccurate in some cases. Confidence scores help filter out nodes with abnormally high rates of bad measurements. When adjusting the tolerance parameter $\tau = 0.2$ and keeping the threshold $\upsilon = 0.2$, two thirds of the nodes have confidence above the threshold. The median localization error for nodes with a confidence score $c_i > \upsilon$ is 55 km, while the median error for nodes with a confidence score $c_i < \upsilon$ is 256 km.

7 Overhead of *VerLoc*

VerLoc is intended for networks that already have a system for broadcasting node descriptors, so that this requirement does

not impose additional overhead as it piggy-backs on existing infrastructure. The network may also collaboratively generate a periodic random beacon for other purposes, or rely on an external source of randomness.

In terms of data broadcast, the overhead caused by *VerLoc* is given by the inclusion of RTT measurements in a blockchain. Each measurement contains an RTT and an identifier of the node being measured, which together can be encoded in 5 B: 20 bit for the node id of the reference being measured (for networks of up to 1 million nodes) and 20 bit for encoding the RTT with a granularity of 1 µs. Considering $R = 40$ references per node (shown to be sufficient in real-world experiments) this amounts to 200 B per node. The overhead scales linearly with the size of the network, e. g. for a network of ten thousand nodes, the broadcast data needed to run *VerLoc* amounts to 2 MB for the full network. If *VerLoc* runs once or twice a day this is a rather small overhead. A resource-efficient version of *VerLoc* can be limited to updating information and measurements only for nodes that have newly joined or that have updated their public key, self-reported location or IP address.

In terms of *VerLoc*'s pairwise communication overhead, each node must send 200 ICMP echo requests to 40 references, meaning it executes 8000 pings, once or twice a day. This overhead remains constant as the network grows and it is in practice negligible for the nodes. Using the measurement data made publicly available, the evaluation scripts can be run locally by everyone to obtain location verification results. Our current (non-optimized) Python scripts take less than 120 ms to compute the localization result for a node with 40 references.

8 Related Work

Delay-Based Geolocation. Different delay-based geolocation techniques [4, 15, 31, 33] exist for use cases like cloud storage [20] or the identification of hidden servers [10]. Such techniques overcome inaccuracies in existing databases [16, 37], and outperform WiFi positioning systems [53] or GPS-based approaches [22]. While prior studies emphasize the resilience of delay-based geolocation against *simple* manipulations [1, 2, 9, 19, 36, 50], *VerLoc* is the first to protect against adversaries in a decentralized setting. Furthermore, prior approaches often rely on central authorities [21, 52]. In contrast to *VerLoc*, they depend on trusted information and do not consider targeted adversarial manipulations. Other network geolocation techniques include proxies [51], focus on achieving street-level granularity [50], or analyze the quality of the information in commercial geolocation databases [18].

Adversarial Localization in Other Contexts. Adversarial interference with location information is also relevant in other contexts. For example, prior work demonstrates privacy attacks that allow an adversary to localize and track mobile users based on public information [17, 28, 43]. In the context of GPS, spoofing attacks are a persisting problem due to the unencrypted and unauthorized transmission of information [23, 45, 47]. Several real-world incidents demonstrate the threat of such attacks [6, 38, 41, 42].

Distance Bounding. Distance bounding protocols are two-party cryptographic protocols that enable a verifier V to establish an upper bound on the physical distance to a (possibly adversarial) prover P [7]. Such protocols are typically designed for bounding distances in the order of metres (or even centimetres [39]) and are mainly concerned with dishonest provers and man-in-the-middle adversaries that try to fake a shorter distance between P and V [35]. In contrast, *VerLoc* is a decentralized multi-party protocol that relies on redundancy across dozens of measurements with randomly selected parties to infer geolocation on a map (rather than just focusing on establishing an upper bound on the distance between two parties), such that results are accurate enough even if some measurements are fabricated by the adversary. An accuracy in the order of tens of kilometres is in most cases good enough for verifying location at country-level granularity in a global or continental network.

9 Conclusion

We have introduced *VerLoc*, a decentralized protocol that uses timing probes between randomly assigned pairs of nodes to verify the geolocation of nodes in a network. *VerLoc* outputs an estimated most likely location, a verification decision on whether the node is in the claimed geographical area (e. g., country), and a score of the confidence of *VerLoc* on a node's localization results. Low scores are indicative of poor network conditions or of active attack.

To configure *VerLoc*, we first conducted an empirical study where we measured real-world propagation timings. We used this information to create a propagation model that summarizes realistic transmission times and speeds as a function of the distance between the nodes. In a series of simulation experiments, we analyzed the performance of *VerLoc* in a non-adversarial setup and further tested its defensive capabilities in increasingly adversarial conditions. Our results show that *VerLoc* protects against false claimed locations even when up to 20 % of nodes are malicious while providing correct location estimates for the remaining benign nodes in 90 % of cases. Finally, we validated *VerLoc* with an experiment in the wild, where several thousand nodes run a *VerLoc* prototype. The real-world results show that *VerLoc* localizes nodes with a median error of 60 km and is thus suitable for verifying locations at country-level granularity.

Acknowledgments

We would like to thank the Nym team for enabling the VerLoc prototype. Being given the opportunity to conduct real-world measurements at this scale and in such a diverse infrastructure

is unique and substantially contributed to the quality of this work. In particular, we would like to thank Dave Hrycyszyn and Jedrzej (a.k.a. "Andrew") Stuczynski for their implementation of the measurements. Additionally, we would like to thank Evgeny Garanin and Sergei Korolev from Nodes Guru for integrating our prototype in their system, and making the localization results of the Nym network publicly available.

This work was in part supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2092 CASA – 390781972, by the Research Council KU Leuven under the grant C24/18/049, by CyberSecurity Research Flanders with reference number VR20192203, and by DARPA FA8750-19-C-0502. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the funders.

A Propagation Model

VerLoc requires a *speed function* to convert measured times to distances, since a simple *speed constant* does not accurately describe the relationship between the physical distance between two computers and the time it takes for data to be transmitted from one to the other via the Internet. The Internet speed function is in fact distance-dependent, with speeds being higher (and noise lower) for longer distances than for shorter ones.

We generate a propagation model with a rather simple method that we introduced in Section 2.2 and that we describe here in more detail. We note that a better propagation model can be generated with a larger and more diverse set of samples. A more sophisticated application of the propagation model can also take into account not just the distance between two nodes, but be finer-grained and account for their actual locations since similar distances can have different speeds depending on the underlying Internet connectivity between locations. While a better propagation model can further improve *VerLoc*'s performance, we have shown in this paper that the *VerLoc* concept works even with a simple model.

We note that building an accurate propagation model benefits from the use of trusted landmarks at known locations. Since this only needs to be done *once* before deploying *VerLoc* and then the model is a publicly known function (that can be used not just by *VerLoc* but by any system that wants to convert internet timings to distances), we argue that this does not introduce trusted parties in *VerLoc*'s operations.

Setup

In order to build the propagation model, we measure the round-trip timings (RTTs) of transmissions between servers in 16 known different worldwide locations and 6042 relays of the Tor network. Between each pair of nodes we send 200 ICMP echo requests to derive the minimum RTT, the mean, and the standard deviation over all pings sent between the two servers. Along with these measured timings, we document the relays' GeoIP locations. Note that while the GeoIP location cannot serve as reliable ground truth, it provides an approximate location for the majority of nodes [18, 51].

Of the 200 timing probes exchanged between a pair of nodes, we take the *minimum* measured RTT as that corresponds to the *least noisy* measurement sample for a connection. The same procedure is followed in *VerLoc* whenever two nodes measure each other: they exchange 200 probes and report the minimum measured RTT as best representative of the channel. This is because noise strictly adds (never subtracts) latency.

Dataset

Within this setup, we conducted a set of 1.8 million measurements within two days, and collected two smaller sets (60k and 30k) weeks later to verify that the RTT distributions were stable. Fig. 1 documents the measurements and the fitting function, which we later use in the simulation and prototype experiments. We sanitized our dataset by removing nodes for which the measurements indicate they announced a false GeoIP entry. In particular, we flag measurements that would imply a propagation speed faster than light. We keep measurements that are (significantly) slower than expected, as this can be caused by network delays. The speed of light is however a hard upper bound for transmissions, and thus nodes with excessively fast measurements can be safely removed.

B Quality of Self-Reported Data

The current prototype implementation of *VerLoc* deployed in Nym uses self-reported locations on a city level. This can only be an approximation of the exact node location, as we refer to the city center for the latitude and longitude of a node. Furthermore, we have no access to ground truth information to check the correctness of the self-reported location. To compensate for this, we compare the city-level information provided by the node operators with the GeoIP information related to the IP addresses of the nodes. Although this does not provide us with a trusted set of locations, it provides a sanity check and a general idea of the quality of the self-reported data.

In total, we test 1395 nodes of our real-world data set. This set represents the number of European nodes for which we successfully fetched GeoIP[8] information. In this comparison, the median location error for all nodes, i. e., the distance between the GeoIP and the self-reported location is 7.11 km with a mean of 364.47 km. To account for the outliers, we then identify all nodes where the self-reported country and the GeoIP country contradict each other. In total we find 126

[8] We use the Maxmind GeoLite 2 City data base, last updated on Sep 09 2021

(9.03 %) with conflicting countries. The median location error here is 1238 km with a mean of 3270 km.

Finally, we revisit the results of the prototype evaluation to identify possible sources of noise leading to verification failures. To this end, we focus on two main characteristics. First, we check for inconsistencies in the self-reported and GeoIP locations. Second, we look for *VerLoc* decisions in which a majority of *slow* timing measurements influenced the confidence score. We do so for both accepted and rejected nodes. Out of 32768 nodes in total (tested in 64 random repetitions), 26993 (82.38 %) nodes were accepted and 5775 (17.62 %) were rejected. Out of the rejected, 1664 (28.81 %) had a conflicting GeoIP location, and 3448 (59.71 %) used measurements of which a majority (more than 80 %) was unexpectedly slow. At the same time, only 832 (3.08 %) of all accepted nodes had a conflicting GeoIP location. Please note that these results can only represent a snapshot. Repeated runs of *VerLoc* allow to monitor node localization and verification over time in order to better assess whether a node is truthfully reporting its location. A node that consistently fails location verification should therefore appear as more suspicious than one that occasionally fails due to slow measurements. Such longitudinal analysis approach is however out of scope for the current version of *VerLoc*.

References

[1] AbdelRahman Abdou, Ashraf Matrawy, and Paul C Van Oorschot. Taxing the Queue: Hindering Middleboxes from Unauthorized Large-Scale Traffic Relaying. *IEEE Communications Letters*, 19(1):42–45, 2014.

[2] AbdelRahman Abdou, Ashraf Matrawy, and Paul C Van Oorschot. Accurate Manipulation of Delay-Based Internet Geolocation. In *ACM Asia Conference on Computer and Communications Security*, AsiaCCS '17, pages 887–898, Abu Dhabi, UAE, April 2017. ACM.

[3] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. LASTor: A Low-Latency AS-Aware Tor Client. In *IEEE Symposium on Security and Privacy*, SP '12, pages 476–490, San Francisco, CA, USA, May 2012. IEEE.

[4] Mohammed Jubaer Arif, Shanika Karunasekera, Santosh Kulkarni, Ajit Gunatilaka, and Branko Ristic. Internet Host Geolocation using Maximum Likelihood Estimation Technique. In *International Conference on Advanced Information Networking and Applications*, AINA '10, pages 422–429, Perth, Australia, April 2010. IEEE.

[5] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 313–314, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[6] Jahshan A. Bhatti and Todd E. Humphreys. Hostile Control of Ships via False GPS Signals: Demonstration and Detection. Technical report, The University of Texas at Austin, 2014.

[7] Stefan Brands and David Chaum. Distance-bounding protocols. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 344–359, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[8] David Canellis. Research: China has the power to destroy Bitcoin, October 2018.

[9] Martin Casado and Michael J Freedman. Peering through the shroud: The effect of edge opacity on ip-based client identification. In *USENIX Symposium on Networked Systems Design and Implementation*, NSDI '07, Cambridge, MA, USA, April 2007.

[10] Claude Castelluccia, Mohamed Ali Kaafar, Pere Manils, and Daniele Perito. Geolocalization of Proxied Services and its Application to Fast-Flux Hidden Servers. In *ACM SIGCOMM Conference on Internet Measurement*, IMC '09, pages 184–189, Budapest, Hungary, July 2009.

[11] Balakrishnan Chandrasekaran, Mingru Bai, Michael Schoenfield, Arthur Berger, Nicole Caruso, George Economou, Stephen Gilliss, Bruce Maggs, Kyle Moses, David Duff, et al. Alidade: IP Geolocation Without Active Probing. *Department of Computer Science, Duke University, Tech. Rep. CS-TR-2015.001*, 2015.

[12] Jingning Chen, Fenlin Liu, Xiangyang Luo, Fan Zhao, and Guang Zhu. A Landmark Calibration-Based IP Geolocation Approach. *EURASIP Journal on Information Security*, 2016(1):1–11, 2016.

[13] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. *ACM SIGCOMM Computer Communication Review*, 34(4):15–26, 2004.

[14] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym Network. https://nymtech.net/nym-whitepaper.pdf, February 2021.

[15] Brian Eriksson, Paul Barford, Joel Sommers, and Robert Nowak. A Learning-Based Approach for IP Geolocation. In *International Conference on Passive and Active Network Measurement*, PAM '10, pages 171–180, Zurich, Switzerland, 2010. Springer.

[16] Brian Eriksson and Mark Crovella. Understanding Geolocation Accuracy Using Network Geometry. In *IEEE Conference on Computer Communications*, INFOCOM '13, pages 75–79, Turin, Italy, April 2013. IEEE.

[17] Dan Forsberg, Huang Leping, Kashima Tsuyoshi, and Seppo Alanärä. Enhancing Security and Privacy in 3GPP E-UTRAN Radio Interface. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2007.

[18] Manaf Gharaibeh, Anant Shah, Bradley Huffaker, Han Zhang, Roya Ensafi, and Christos Papadopoulos. A Look at Router Geolocation in Public and Commercial Databases. In *ACM SIGCOMM Conference on Internet Measurement*, IMC '17, pages 463–469, London, UK, November 2017. ACM.

[19] Phillipa Gill, Yashar Ganjali, Bernard Wong, and David Lie. Dude, where's that IP?: Circumventing Measurement-Based IP Geolocation. In *USENIX Security Symposium*, USENIX '10, pages 16–16, Washington, DS, USA, August 2010. USENIX.

[20] Mark Gondree and Zachary NJ Peterson. Geolocation of Data in the Cloud. In *Conference on Data and Application Security and Privacy*, CODASPY '13, pages 25–36, San Antonio, TX, USA, 2013. ACM.

[21] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-Based Geolocation of Internet Hosts. *IEEE/ACM Transactions on Networking (TON)*, 14(6):1219–1232, 2006.

[22] Dexter H Hu and Cho-Li Wang. GPS-Based Location Extraction and Presence Management for Mobile Instant Messenger. In *International Conference on Embedded and Ubiquitous Computing*, pages 309–320. Springer, 2007.

[23] Todd E. Humphreys, Brent M. Ledvina, Mark L. Psiaki, Brady W. O'Hanlon, and Paul M. Kintner Jr. Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer. In *International Technical Meeting of the Satellite Division of The Institute of Navigation*, ION GNSS '08, pages 2314–2325, Savannah, GA, USA, September 2008.

[24] Kai Jansen, Matthias Schäfer, Daniel Moser, Vincent Lenders, Christina Pöpper, and Jens Schmitt. Crowd-GPS-Sec: Leveraging Crowdsourcing to Detect and Localize GPS Spoofing Attacks. In *IEEE Symposium on Security and Privacy*, SP '18, pages 1018–1031, San Francisco, CA, USA, May 2018. IEEE.

[25] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards IP Geolocation Using Delay and Topology Measurements. In *ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 71–84, Rio de Janeiro, Brazil, October 2006. ACM.

[26] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 357–388, Cham, 2017. Springer International Publishing.

[27] Katharina Kohls, Kai Jansen, David Rupprecht, Thorsten Holz, and Christina Pöpper. On the Challenges of Geographical Avoidance for Tor. In *Network and Distributed System Security Symposium*, NDSS '19, San Diego, CA, USA, February 2019. The Internet Society.

[28] Katharina Kohls, David Rupprecht, Thorsten Holz, and Christina Pöpper. Lost Traffic Encryption: Fingerprinting LTE/4G Traffic on Layer Two. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 249–260, 2019.

[29] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In *IEEE Symposium on Security and Privacy*, SP '18, pages 583–598, San Jose, CA, USA, 2018. IEEE.

[30] Dan Komosny, Milan Simek, and Ganeshan Kathiravelu. Can Vivaldi Help in IP Geolocation? 2013.

[31] Sándor Laki, Péter Mátray, Péter Hága, Tamás Sebők, István Csabai, and Gábor Vattay. Spotter: A Model Based Active Geolocation Service. In *International Conference on Computer Communications*, INFOCOM '11, pages 3173–3181, Shanghai, China, April 2011. IEEE.

[32] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[33] Dan Li, Jiong Chen, Chuanxiong Guo, Yunxin Liu, Jinyu Zhang, Zhili Zhang, and Yongguang Zhang. IP-Geolocation Mapping for Moderately Connected Internet Regions. *Transactions on Parallel and Distributed Systems*, 24(2):381–391, 2012.

[34] Zhihao Li, Stephen Herwig, and Dave Levin. DeTor: Provably Avoiding Geographic Regions in Tor. In *USENIX Security Symposium*, USENIX '17, pages 343–359, Vancouver, BC, Canada, August 2017. USENIX Association.

[35] Aikaterini Mitrokotsa, Cristina Onete, and Serge Vaudenay. Mafia fraud attack against the rČ distance-bounding protocol. In *2012 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*, pages 74–79, 2012.

[36] James A Muir and Paul C Van Oorschot. Internet Geolocation: Evasion and Counterevasion. *Acm computing surveys (csur)*, 42(1):1–23, 2009.

[37] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. IP Geolocation Databases: Unreliable? *ACM SIGCOMM Computer Communication Review*, 41(2):53–56, 2011.

[38] Mark L. Psiaki and Todd E. Humphreys. Attackers can spoof navigation signals without our knowledge. Here's how to fight back GPS lies. *IEEE Spectrum*, 53(8):26–53, August 2016.

[39] Kasper Bonne Rasmussen and Srdjan Capkun. Realization of rf distance bounding. In *Proceedings of the 19th USENIX Security Symposium*, pages 389 – 401, Washington, DC, 2010. USENIX Association. 19th USENIX Security Symposium 2010; Conference Location: Washington, DC, USA; Conference Date: August 11-13, 2010.

[40] Florentin Rochet, Ryan Wails, Aaron Johnson, Prateek Mittal, and Olivier Pereira. Claps: Client-location-aware path selection in tor. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 17–34, New York, NY, USA, 2020. Association for Computing Machinery.

[41] Mary-Ann Russon. Wondering how to hack a military drone? It's all on Google, May 2015.

[42] Clare Sebastian. Getting lost near the Kremlin? Russia could be 'GPS spoofing', December 2016.

[43] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, and Jean-Pierre Seifert. Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. In *Network and Distributed System Security Symposium*. The Internet Society, 2016.

[44] Yuval Shavitt and Noa Zilberman. A Geolocation Databases Study. *IEEE Journal on Selected Areas in Communications*, 29(10):2044–2056, 2011.

[45] Peter F. Swaszek and Richard J. Hartnett. Spoof Detection Using Multiple COTS Receivers in Safety Critical Applications. In *International Technical Meeting of The Satellite Division of the Institute of Navigation*, ION GNSS+ '13, pages 2921–2930, Nashville, TN, USA, September 2013.

[46] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable Bias-Resistant Distributed Randomness. In *IEEE Symposium on Security and Privacy*, SP '17, pages 444–460, San Jose, CA, USA, 2017. IEEE.

[47] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Čapkun. On the Requirements for Successful GPS Spoofing Attacks. In *ACM Conference on Computer and Communications Security*, CCS '11, pages 75–86, Chicago, IL, USA, October 2011. ACM.

[48] Carmela Troncoso, George Danezis, Marios Isaakidis, and Harry Halpin. Systematizing decentralization and privacy: Lessons from 15 years of research and deployments. *CoRR*, abs/1704.08065, 2017.

[49] S. Čapkun and J. P. Hubaux. Secure Positioning of Wireless Devices with Application to Sensor Networks. In *IEEE Conference on Computer Communications*, INFOCOM '05, pages 1917–1928, Miami, FL, USA, March 2005. IEEE.

[50] Yong Wang, Daniel Burgener, Marcel Flores, Aleksandar Kuzmanovic, and Cheng Huang. Towards Street-Level Client-Independent IP Geolocation. In *USENIX Symposium on Networked Systems Design and Implementation*, NSDI '11, pages 27–27, Boston, MA, USA, March 2011. USENIX Association.

[51] Zachary Weinberg, Shinyoung Cho, Vyas Sekar, and Phillipa Gill. How to Catch when Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation. In *ACM SIGCOMM Conference on Internet Measurement*, IMC '18, Boston, MA, USA, October 2018. ACM.

[52] Bernard Wong, Ivan Stoyanov, and Emin Gün Sirer. Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In *USENIX Symposium on Networked Systems Design and Implementation*, NSDI '07, pages 23–23, Santa Clara, CA, USA, June 2007. USENIX Association.

[53] Paul A. Zandbergen. Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning. *Transactions in GIS*, 13:5–25, 2009.

[54] Wolfie Zhao. Top Bitcoin Mining Pools See 15% Hashrate Drop Amid Continuous Rainstorms in China, August 2020.