Certified Internet Coordinates

Mohamed Ali Kaafar*, Laurent Mathy †, Chadi Barakat*, Kavé Salamatian †, Thierry Turletti * and Walid Dabbous * *INRIA, France †Lancaster University, UK

***Abstract*—We address the issue of asserting the accuracy of coordinates advertised by nodes of Internet coordinate systems during distance estimations. Indeed, some nodes may lie deliberately about their coordinates to mount various attacks against applications and overlays. Our proposed method consists in two steps: 1) establish the correctness of a node's claimed coordinate (which leverages our previous work on securing the coordinates embedding phase using a Surveyor infrastructure); and 2) issue a time limited validity certificate for each verified coordinate. Validity periods are computed based on an analysis of coordinate inter-shift times observed on PlanetLab, and shown to follow a long-tail distribution (lognormal distribution in most cases, or Weibull distribution otherwise). The effectiveness of the coordinate certification method is validated by measuring the impact of a variety of attacks on distance estimates.**

I. Introduction

Internet coordinate systems, or ICS in short (e.g. [1], [2], [3]) have been proposed to allow for delay (Round-Trip Time, shortly RTT, or distance) estimation between nodes, in order to reduce the measurement overhead of many applications and overlay networks. Indeed, by embedding the Internet delay space into a metric space – an operation that only requires each node in the system to measure delays to a small set of other nodes (called neighbors) –, nodes are attributed coordinates that can then be used to estimate the RTT between any two nodes, without further measurements. Simply by applying the distance function associated with the chosen metric space to the nodes' coordinates, every RTT between nodes participating in the Internet coordinate system, is estimated.

The security of the embedding phase, or in other words the coordinate computation phase, of Internet coordinate systems has received some attention in the form of either simple mechanisms built into embedding protocols [1], [2] or more general and independent cheat detection mechanisms [4], [5], [6]. All of these mechanisms use distance measurements between two nodes to assess the plausibility of the corresponding distance estimation based on the nodes' current coordinates. In cases where the discrepancy between the measured and estimated distances is deemed too important, the node carrying out the test removes its correspondent node from its set of neighbors. By doing this, it avoids adjusting its coordinate in response to potentially malicious information.

However, ultimately, ICS are used to estimate distances between nodes, based on their coordinates only, even and all the more so if these nodes have never exchanged a distance measurement probe. Whatever mechanism is used to obtain a node's coordinate (gossip-based exchange, DNS-like repository, etc.), each node must somehow report its own coordinate computed during the embedding phase of the system. This is because, for scalability reasons, nodes compute their own coordinates in a distributed way. A malicious node possesses then a valuable opportunity to strike: in order to achieve some application-dependent goal or advantage (e.g. free-riding, denial-of-service, isolation, ubiquity gift, etc), a node can repeatedly lie about its coordinate. This urges for a solution to verify that nodes use their right coordinates at the application level. Securing the embedding phase is then necessary but still not sufficient in this regard.

Several applications are today concerned with the use of coordinates as provided by ICS. Examples of such applications range from optimizing P2P networks, to the deployment of ICS to offer better performances to Anonymity Networks such as Tor [7]. To illustrate the impact of simple cheating, we consider the application of selecting the closest download server as a potential use of the coordinate systems in the Internet. Each client thus needs to order a set of download servers according to their distances. When requested, corrupted servers provide biased coordinates that are only a small distance away (e.g. 10ms) from the requesting node. We measured the percentage of clients for which corrupted servers manage to break into the top 5 closest servers for these clients (when in reality they are not). We observed that even a small subset of the servers being corrupted could easily lure a noticeable fraction of clients: for instance, even 1% of the servers could erroneously attract more than 20% of clients.

Through this particular application, we can see the effectiveness of obvious attacks consisting in simply lying about coordinates. Several other studies have also quantified the impact of cheating on topology-aware Internet applications, and have shown that simple attack strategies can prove very effective [8], [9]. In this context, this paper addresses the question of guaranteeing the veracity of the coordinates advertised by nodes. To do so, we propose to leverage the Surveyor infrastructure and embedding cheat detection test proposed in [4]. More precisely, using such detection test, we propose that a few trusted entities called Surveyors, measure their distance to a node in order to verify the correctness of its claimed coordinate. If all Surveyors agree that this coordinate is the node's true coordinate, a time-limited validity certificate, including the certified coordinate, is issued to the node.

A certificate validity time period is necessary because, due to dynamic network conditions, nodes' coordinates vary naturally in time. Upon a coordinate change, an honest node would stop using its current certificate and seek a certification of its new coordinate. On the other hand, a malicious node would keep using a certificate related to a previous position, hence a careful balance between scalability and certificate validity is desirable. To achieve this, one of our contributions is to study the coordinate inter-shift time (i.e. the time between

coordinate changes at a node) as observed for a Vivaldi system running on PlanetLab. We found that the coordinate inter-shift times at most nodes follow a lognormal distribution, with the rare cases when this distribution is inappropriate being accounted for by a Weibull distribution (note these are both long-tail distributions). We leverage this observation to derive optimal validity periods for certificates.

The remainder of the paper is as follows: in Section II, we study and characterize the coordinate inter-shift times and show that these times observed at Surveyors can adequately and statistically model inter-shift times at nearby nodes. Section III describes the certification procedure in detail, while performance evaluation of our proposal in the context of various attacks is presented in Section IV. Section V concludes the paper.

II. Coordinates Evolution model

To model the coordinate evolution, we concentrate on tracking the evolution in time of coordinates along the different dimensions of the coordinate space by observing their evolution in a clean system (without any malicious activity).

A. Experimental Set-up

Our studies rely on both off-line measurements traces and live-experiments. We used the traces to study Internet coordinate systems dynamics and used live-PlanetLab experiments, as a Vivaldi service deployed over a three-weeks period, to demonstrate the validity and effectiveness of our proposal to deal with various attacks.

First, the traces were obtained by running the Vivaldi System on 450 PlanetLab machines through a period of 18-days. These traces were then acquired in a clean environment with no malicious node. Vivaldi uses a 3-dimensional Euclidean space for the embedding. Each 10 minutes, corresponding to an embedding step, nodes are adjusting their coordinates based on a one-to-one interaction with one of its neighbors.

On the other hand, the live-PlanetLab experiments were conducted during 14-days over a set of 280 PlanetLab nodes spread world-wide, running Vivaldi as a coordinate-embedding service [1]. For the purpose of our experimentations, we slightly modified the logging functions of the Vivaldi protocol. Each node is running several instantiations to allow us experimenting different parameters in similar conditions. Nodes are then updating their coordinates as needed, depending on the embedding step defined in each instantiation of the Vivaldi protocol. In the same way, the behavior of nodes, acting as malicious nodes or as honest 'normal' nodes varies from one instantiation to another. Each node has 20 neighbors (i.e. attached to 20 springs), 10 of which being chosen to be closer than 50 ms. The number neighbors was actually chosen because in our experiments Vivaldi performs as good with 32 neighbors (as recommended in [3]) than with 20 neighbors. The constant fraction C_c for the adaptive timestep (see [3] for details) is set to 0.25. When needed, Surveyor nodes were chosen randomly and represent 8% of the overall population [4].

[1]The number of nodes in the live-experiments was reduced because of availability of nodes on PlanetLab

B. Observations

As we observed similar coordinates evolution in both our off-line traces and live-experiments, we focus in this section on the results as observed in our traces (observations are done during longer periods). Figure 1 shows a typical evolution of the coordinates of a Vivaldi node. Each sub-figure depicts the evolution of one coordinate component along one of the three dimensions($dim1$, $dim2$ and $dim3$) used for the coordinates embedding.

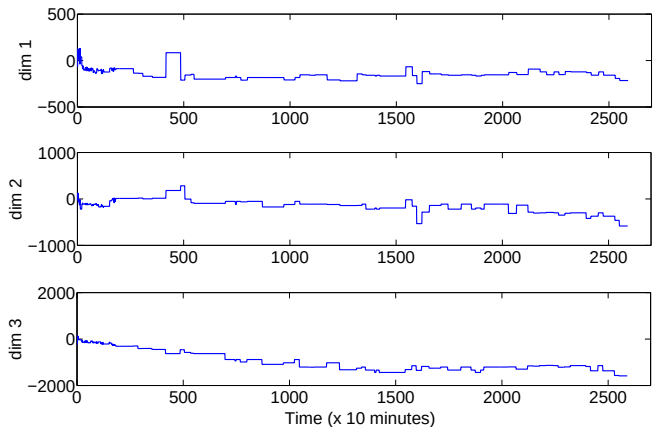


Fig. 1. Typical Variations of a node's coordinate in the Vivaldi System.

We observe that the system after roughly 250 embedding steps, reaches a stationary regime, but coordinates continue to fluctuate. Looking further in the entire set of nodes' coordinates, we observe that the rate of these variations is low enough to allow distance estimation at the application level, but this unfortunately prevents us from certifying absolute (permanent) coordinates.

More specifically, because at any instant in time, the RTT that can be measured between two nodes depends on the state of the network (e.g. traffic load, state of queues in routers, etc), the exact value of the RTT varies continuously. However, it has been shown that RTT values in the Internet exhibit some stability in a statistical sense [12], with the statistical properties of RTTs exhibiting no significant change at timescales of several minutes. It is that property that embedding systems exploit to provide good distance estimates while only needing to have nodes adjust (recalculate) their coordinates on a periodic basis. From modeling point of view, the coordinate of a node can be viewed as a discrete stochastic process, embedded at the instants of updates.

Regardless of the dimensionality used by the coordinate-systems, our main goal is to assign to any coordinate given by a node, a reliability value that is telling the likelihood that this coordinate is still valid and has not changed. For this purpose we observe the inter-shift time distribution, corresponding to the amount of time (in terms of embedding steps intervals) during which, nodes stick to their positions, i.e. the coordinates do not change. This distribution is denoted T_i for each node i. It is important to note that although we observed that in our traces, a variation of one coordinate component was synonym to the variation of both others, we consider the inter-shift time

as the laps of time corresponding to the non variation of all the coordinate components of this node. Basically, we would like to determine which probability distribution is suitable to describe the inter-shift times. In the following, we will use our empirical data sets of inter-shift times to find the probability distribution that best describes the distribution values of each T_i.

C. Inter-shift Time Distribution Fitting

For choosing the best suitable distribution, we use a set of candidate distributions containing lognormal, Weibull, Rayleigh and Gamma distributions [2]. For each node in the dataset, we apply a two-step procedure. In the first step, we derive the maximum likelihood estimates of parameters of each distribution in the candidate set. The likelihood is the probability to get the observed inter-shift times for some hypothetic distribution. The estimates of the distribution parameters are then the values that maximize their likelihood. In a second step, we used goodness of fit tests to evaluate if the hypothesis that the observed values T_i come from the candidate distribution can be rejected or not. The goodness of fit evaluation was done using the popular and robust Kolmogorov-Smirnov (K-S) test [13], applied at a significance level of 5%.

Using the fitting procedure described above, we tried to fit the inter-shift datasets. The first interesting result we found is that all of the empirical distributions examined can be fitted to a known distribution. A large majority of distributions can be fitted into a lognormal distribution. The lognormal hypothesis was rejected for only 5 datasets out of the 450. Looking further in these 5 inter-shift datasets, we observed that they have a good fit with the Weibull distribution. Table I gives a summary of our findings for the Kolmogorov Smirnov goodness of fit tests.

D. Correlation between Surveyors and Nodes inter-shift Distributions

Having shown that inter-shift time distribution of our population of nodes in the Vivaldi system is compatible with either a lognormal distribution in most cases or a Weibull distribution otherwise whose parameters can be obtained by a maximum likelihood method, the next question is how well the shifts as observed by Surveyor nodes can be used as representative of the shifts of regular nodes. Basically, if the inter-shift distribution as seen by the surveyors is the same as the real inter-shift distribution of regular nodes, the former may be used as a reference to validate the node's coordinate.

The verification of this hypothesis is done by comparing the sequence of inter-shift times as seen by surveyors and the real inter-shift of a node and asking if the hypothesis that these two sequences come from the same distribution can be rejected or not. The latter test is done using a two-sample Kolmogorov Smirnov Test (with a significance level of 5%) that precisely gives an answer to the previous question. For each of the 35 surveyor nodes, we applied therefore 415 two-sample K-S test. We then analyzed the likelihood that the two-samples KS-test is rejected as a function of the distance (measured as an RTT) between the surveyor node and the tested regular node. The likelihood is obtained as the ratio between the number of nodes with a given RTT that reject the test and the overall number of regular nodes. Figure 2 shows this rejection ratio *vs.* the distance (measured as an RTT) between a node and the corresponding Surveyor, as observed during the PlanetLab experiment.

[2]The Gaussian distribution was not tested because the empirical distribution was not symmetrical around a mean.

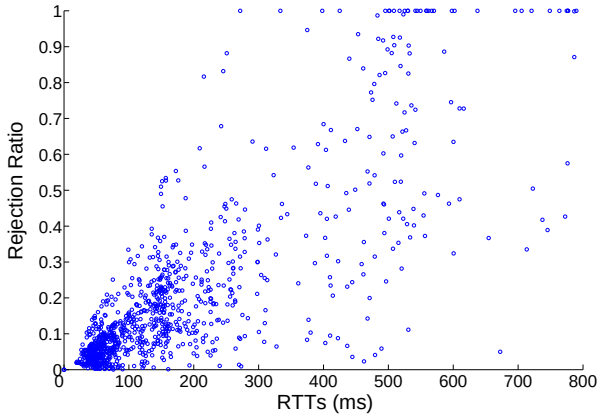


Fig. 2. Correlation between 'Nodes-Surveyors' RTTs and the rejection Ratio.

Intuitively, a Surveyor should have tendency to have the same inter-shift distribution as nodes that are close by in terms of RTT, as they are more likely to experience similar dynamics of the coordinate system. Figure 2 validates this intuition and shows that better locality between a node and its Surveyor yields more accurate fittings. We therefore apply the following heuristics: the inter-shift time distribution of the closest Surveyor is used as the representative distribution for regular nodes.

III. Coordinate Certification

The method we propose to certify Internet coordinates consists in two steps:

1) the node coordinate verification test;
2) computation of an estimated validity period for this coordinate.

The coordinate verification test leverages the Surveyor infrastructure and malicious embedding neighbor detection proposed in [4], while the validity period estimation is based on the results presented in section II.

A. Coordinate Verification

1) Principle: Before going into the details of our coordinate verification protocol, let's first describe briefly the malicious embedding detection test, on top of which we build our protocol.

The detection test is based on a model for the evolution of nodes' observed relative errors updated at embedding steps. Because of the linear properties of the model, a Kalman filter

TABLE I
RESULTS USING THE KOLMOGOROV SMIRNOV TEST FOR FITTING THE INTER-SHIFT TIMES DATA.

Fitted Distributions	% of samples that passed the test
lognormal	445/450
Rayleigh	2/450
Weibull	5/450
Gamma	6/450

can be used to track the evolution of these relative errors and to predict their values in the future steps. The main strategy behind the malicious node embedding detection is that if the stochastic space model, and especially its associated Kalman filter, are calibrated within a clean embedding system, then a simple hypothesis test can be used to assess whether the deviation between the measured relative error and the predicted relative error, observed at a given embedding step, is normal or is the fruit of anomalous or malicious activity from the peer node. The idea is thus to have a set of trusted nodes (a.k.a Surveyors), well positioned in the network and acting as vantage points, that compute their coordinates by embedding with each other exclusively and observe the evolution of their relative errors in a clean system without influence from malicious nodes. The Kalman filters calibrated on such nodes can advantageously track the relative error model at nearby untrusted nodes. Each Kalman filter is capable of providing a predicted relative error (based on the history of observed measured errors) as well as the distribution of its innovation process which is the deviation between the measured and predicted relative errors expected at each embedding step. The basis for the malicious embedding neighbor detection test is then a simple hypothesis test on this innovation process.

Thus, this test simply verifies the consistency between the estimated and measured distances between two nodes. In particular, this test is not able to evaluate the truthfulness of a node's coordinate. Indeed, if during an embedding step a node fakes its coordinate but at the same time delays the measurement probes in a way consistent with its faked position in the coordinate space (based on the knowledge of the correspondent's coordinate, as well as its own true and fake coordinates), then the resulting relative error measured between the two nodes is reduced. Therefore, a node faking a position further away from the testing node will never fail a test that it wouldn't have failed if it wasn't faking.

Consequently, to verify a node's coordinate, several such tests must be performed from vantage points (Surveyors) *surrounding* the node. In this case, a node could easily fake its coordinate and consistently delay probes so that it moved away from some Surveyors without being noticed. But such fake position would necessarily also result in the node moving closer to some other Surveyors and failing the corresponding malicious embedding neighbor tests as it is next to impossible to "speed up" a distance probe protected by the simplest of mechanisms (e.g. hashing, simple encryption, random probe numbers, etc). A node must thus be surrounded by at least one more Surveyors than there are dimensions in the space.

If the malicious embedding neighbor test is negative at each Surveyor chosen to surround the node (i.e. the relative error observed between the Surveyor and the node is considered normal), then the coordinate claimed by the node is considered correct. Note that this test is different from a normal "triangulation" approach, where the measured distances between the node and the Surveyors would be used alongside the Surveyors' coordinates to determine the node's own coordinate. Indeed, our test is in fact made up of multiple, independent tests on the plausibility of the observed relative errors and provides our method with an important resistance to the triangular inequality violations (TIVs) [14], [15] that can be commonly encountered in the Internet. This is because the Kalman filters underlying the tests are calibrated during normal embedding of the Surveyors, and thus in conditions where TIVs are naturally encountered, so the system noise resulting from these TIVs is therefore implicitly taken into account in the relative error tracking. We do not claim that our test is immune to the problems caused by TIVs (and these TIVs will be responsible for some of the false positives of our test), but it is nevertheless much less sensitive to them than a geometric approach like triangulation would be.

2) Protocol: A node who wishes to have its coordinate certified contacts the known closest Surveyor to its claimed coordinate. If this Surveyor is not the closest, the node is redirected to the closest one known by the Surveyor. For this, as well as the selection of Surveyors surrounding the coordinate claimed by a node to happen, Surveyors exchange their coordinates using a gossiping protocol.

Based on its knowledge of the position of other Surveyors, as well as on the coordinate of the node to be certified, the certifying Surveyor selects a set of Surveyors (a.k.a surrounding Surveyors) that surround the node's claimed position (possibly including itself). Then it informs these of the node's claimed coordinate so to ensure they all use the same node's coordinate for their distance estimates during their malicious embedding neighbor detection test.

Note that Surveyors compute their own coordinate by using each other exclusively as embedding neighbors. This gives the Surveyors the view of a clean system without malicious insider attacks. Therefore, if Surveyors run their own malicious embedding neighbor detection test at each embedding step, all such tests should ideally be negative as Surveyors only have trusted and honest neighbors (other Surveyors). Unfortunately, no test is perfect and some amount of the tests carried out by each Surveyor will wrongly identify the neighbor as malicious. Such occurrence constitutes a false positive and the Surveyor will take no action about the said neighbor. However, carrying out such tests at every embedding step provides the Surveyors

with estimates of two important test statistics: the false positive test ratio ($FPTR$ – i.e. the percentage of the tests that were positive and wrongly identified a fellow Surveyor as malicious) and the true negative test ratio ($TNTR$ – i.e. the percentage of the tests that were negative and thus correctly identified the fellow Surveyors as honest nodes).

Let Y_i be the indicator random variable that represents the outcome of a malicious embedding neighbor detection test at the i^{th} Surveyor (testing another Surveyor), with:

$$Y_i = \begin{cases} 0 & \text{if the neighbor is identified as honest} \\ 1 & \text{if the neighbor is identified as malicious} \end{cases}$$

Taking as null hypothesis H_0 that the tested node is honest (which is always the case when Surveyors are tested), the true negative test ratio ($TNTR$) estimate p_i at the i^{th} Surveyor is $\mathbb{P}\mathrm{rob}\{Y_i = 0|H_0\}$, the number of tests that were correct divided by the overall number of tests carried out. The $FPTR$ is then obviously $1 - p_i$.

After performing the requested malicious embedding neighbor detection test on the node whose coordinate is to be certified, the surrounding Surveyors return the result of their test, along with their estimated $TNTR$, to the certifying Surveyor. If every test returned is negative (i.e. each surrounding Surveyor considered the node as honest), then the node's coordinate is deemed correct and verified and the certifying Surveyor proceeds to the second step of the certification described in section III-B.

On the other hand, if at least one of the surrounding Surveyor returns a positive test, that is, did consider the node as potentially malicious because of too much deviation between the measured relative error and the expected one, the certifying Surveyor must decide whether to declare the node's coordinate as suspicious and thus refuse to issue a certificate, or whether further tests should be carried out. To do so, the probability that the node, and its claimed coordinate, have been identified mistakenly as suspicious by the surrounding Surveyors is computed. This probability is simply $1-\prod_{i\in\xi^j} p_i$, where ξ^j is the set of surrounding Surveyors chosen to verify the claimed coordinates of the node at this round of testing. If the overall probability that the node has been mistakenly classified as suspicious is greater than a given significance value γ, that is if $\prod_{1\leq j\leq\mathcal{N}}(1 - \prod_{i\in\xi^j} p_i) > \gamma$, where $\mathcal{N}$ is the number of test rounds that have been carried out so far, then the certifying Surveyor starts another test round with a new set of surrounding Surveyors. Note that the sets of selected surrounding Surveyors at each round are not necessarily disjoint, although such property is desirable. In this paper, we used $\gamma = 1\%$ and limited $\mathcal{N}$ to 6 (i.e. a node is refused a coordinate certificate if the probability of mistaken refusal falls below 1% and/or the node fails 6 consecutive test rounds).

3) Evaluation: In this section, we seek to evaluate the effectiveness of our proposed coordinate verification method. We first validate the assumption that the $FPTR$ values measured during embedding at Surveyors provide good estimates for the real $FPTR$ values at these Surveyors. To do so, we let a Vivaldi system, without cheat, converge and run on PlanetLab for over 2500 time ticks (i.e. embedding periods). The Surveyors measured their estimated $FPTR$ by carrying out a malicious embedding neighbor detection test at every embedding step. At the end of the experiment, each Surveyor also measured its real $FPTR$ by running a malicious embedding neighbor detection test to every other nodes in the system. Since this system does not have any malicious node in it, a failed test is a false positive. The CDF of the differences of these 2 values at each Surveyor is shown in figure 3. We see that the difference between the estimated and the real $FPTR$ are mostly within less than 1% of each other, confirming that our proposed estimation method yields reliable $FPTR$ estimates. Even in the cases where the $FPTR$ estimates differ more than the real value, these will only affect the coordinate verification tests in which the corresponding Surveyors take place: in these cases the coordinate verification test will be slightly more aggressive than it ought to (since the $FPTR$ estimate is smaller than the real value), favoring security.

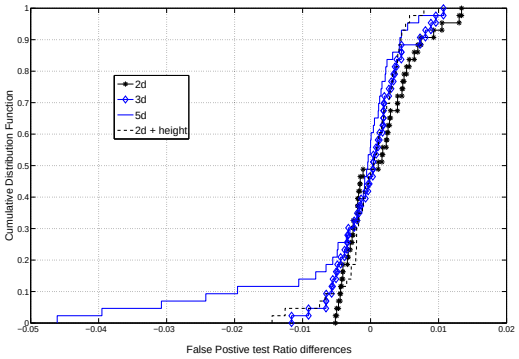


Fig. 3. CDF of False positive probability differences.

Next, we seek to further understand the behavior of false positive and true negative occurrences. We considered then scatter plots of the distance between a node and its Surveyor, whenever the malicious embedding neighbor detection test yields a false positive at the Surveyor. Due to space constraints, we omit those figures, that clearly indicate that Surveyors hardly ever experience wrong test results when testing nearby honest nodes. Complementarily, figure 4 shows that Surveyors are much more successful at identifying nearby honest nodes correctly. These results indicate that striving to choose surrounding Surveyors as close as possible to the node whose coordinate are being verified will increase the effectiveness of the coordinate verification test (by reducing the occurrences of false positive in the multiple test, through reduction of false positive occurrences in the component tests making this multiple test up). This is therefore the strategy adopted in the rest of this paper.

Finally, we experimented with a simple attack, carried out by a growing malicious node population that has access to the coordinates of all nodes in the system. The malicious nodes compute the centroid of the overall node population, and then try to move in a random direction away from this centroid (adding 2 seconds) in order to be isolated. Figure 5(a) shows the detection rate, that is the percentage of certificate requests for faked coordinates that was denied, as a function

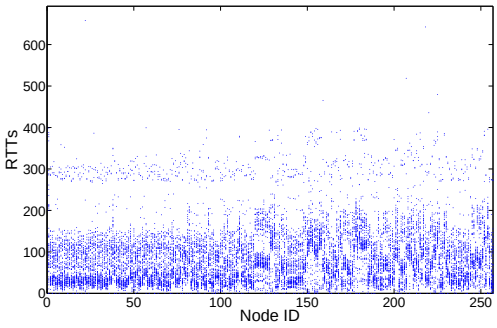
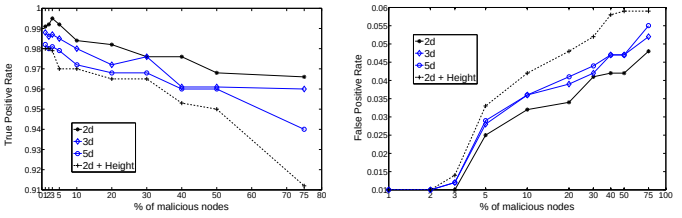


Fig. 4. Correlation between True Negatives and RTTs

of the malicious population size, for various dimensions of the coordinate space. Note that although these curves show a slightly decreasing trend, a smaller percentage of an increasing number of requests for faked coordinates does mean, in most cases, an increasing number of denials. With over 95% detection rates in most cases, the coordinate verification test can be considered as highly efficient.



(a) Self isolation: Detection probability (b) Self Isolation: False Positive Ratio

Fig. 5. Detection Performance

However, tests may achieve high detection rates by being overly aggressive towards honest nodes that do not cheat. The false positive rate, that is the percentage of certificate requests for real coordinates that were wrongly denied, is a measure of this aggressiveness towards non malicious nodes and should be kept as low as possible. Figure 5(b) shows the measured false positive rate, as a function of the size of the malicious population, for various dimensionality of the coordinate space. Note that these curves depend on the performance of the test only, and not on the activities of the malicious nodes. Also, note that as the population of malicious nodes increases, the corresponding population of honest nodes decreases, so an upward trend actually corresponds to fewer wrongly denied certification requests to honest nodes. With a false positive rate lower than 6% in all cases, our test can be considered as moderately, and acceptably aggressive. In light of the evaluation results presented in this section, we conclude that our proposed test for coordinate verification exhibits good performance and is fit for purpose.

B. Certificate Validity Computation

After the correctness of a node's advertised coordinate has been asserted, the next step is to issue the node with a certificate proving that the enclosed coordinate has been verified. This certificate will be delivered by the certifying Surveyor (i.e. usually the Surveyor closest to the node in the coordinate space, see sections II-D and III-A2).

As a coordinate certificate is associated with a particular position in the coordinate space that the node occupies or has occupied, one could expect that nodes request new certificate on moving to a new position. While this is certainly the behavior anticipated from honest nodes that are interested in being able to prove their true position, malicious nodes might decide not to do so. Indeed, a malicious node is probably much more interested in faking its true position, and doing so with a certificate "proving" a position which isn't really its own, whatever this coordinate might be, is probably a bonus for such a node. While the coordinate verification protocol described in section III-A2 has been designed to prevent, as much as possible, malicious nodes from acquiring certificates for fake coordinates that they may choose, the problem of nodes using certificates for positions that have meanwhile changed is still to be addressed.

The obvious way to solve this "stale" certificate problem, is to assign a reliability to the certificate. The reliability of the certificate decreases with time from its issuance time. When it crosses a certain threshold p_{th}, the certificate should be invalidated and eventually reissued. There is a tradeoff between certificate precision (and therefore security) and frequency of coordinate certification that is controlled by the reliability p_{th}. Using larger value of p_{th} leads to higher reliability for certificates, but at the cost of frequently reissuing certificates that are still valid. On the other hand, lower p_{th} results in lower reliability, but also reduces the load on Surveyors who would receive certificate requests less frequently. The issue here is really to find the right trade-off between scalability (by limiting the rate or amount of certificate requests) and security (by limiting the time lapse during which a malicious node may be able to use an old certificate after its coordinate has changed). We will therefore seek to exploit the results on coordinate inter-shift times presented in section II-C to compute certificate validity periods.

Note that this section assumes that all nodes in the system are "loosely" time synchronized: since coordinate inter-shift times have been shown to take values that are usually measured in minutes (see section II), as long as all clocks are synchronized with an accuracy exhibiting a smaller time-scale (say a few seconds), the notion of time in the system can then be considered unique. This time synchronization can easily be obtained through using NTP (at least among surveyors).

1) Principle: The problem of computing a validity period for coordinate certificates can be formalized through reliability theory. Let's define the survival function of the coordinate of node i as the probability, $S^i_{T_0}(\Delta) = \mathbb{P}\text{rob}\{T^i > T^i_0 + \Delta\}$, that the next change of coordinate occurs later than Δ time units after the last coordinate change, happening at time T^i_0. The survival function is thus related to the inter-shift time cumulative distribution $F^i(\Delta) = \mathbb{P}\text{rob}\{T^i \leq T^i_0 + \Delta\}$ through $S^i(\Delta) = 1 - F^i(\Delta)$. Recall from section II-D, that the inter-shift times observed at a Surveyor are similar to those observed at nearby nodes. Hence, the inter-shift time distribution at a certifying Surveyor is the distribution used to compute the

validity of the certificates it issues (since it issues certificates to the nodes that are closest to it than to any other Surveyors).

The survival function can be used to compute the validity of a certificate, that is the time remaining until the next coordinate change. The probability that the next position change occurs at or before time $\tau + \Delta$, given that the certificate is being issued at time τ is:

$$\begin{aligned}\mathbb{P}\text{rob}\{T < \tau + \Delta | T > \tau\} &= \frac{\mathbb{P}\text{rob}\{\tau < T < \tau + \Delta\}}{\mathbb{P}\text{rob}\{T > \tau\}} \\ &= 1 - \frac{S^i(\tau + \Delta - T_0^i)}{S^i(\tau - T_0^i)}\end{aligned}$$

In fact, we use the above probability, computed at a Surveyor whose survival function and last coordinate change time are $S^i(\Delta)$ and T_0^i respectively, to estimate the lapse of time until the next position change of the node requesting the certificate. In other words, the assumption here is that network conditions for nodes that are close to each other should change in a synchronous way. However, due to the asynchronous nature of embedding steps at different nodes, their respective coordinates will not all change at the same time, but we take as "reference" time, the moment when the Surveyor is expected to see a change in its coordinate.

In section II-C, we showed that most nodes follow a lognormal or a Weibull distribution. Depending on which distribution each surveyor node is observing (computing the likelihood at each embedding step), the above formula has a simple form for these two distributions.

For lognormal inter-shift distribution we will have:

$$\begin{aligned}S(\Delta) &= 1 - \Phi(\frac{\ln \Delta}{\sigma}) \\ \mathbb{P}\text{rob}\{T < \tau + \Delta | T > \tau\} &= 1 - \frac{1 - \Phi(\frac{\ln(\tau + \Delta - T_0^i)}{\sigma})}{1 - \Phi(\frac{\ln(\tau - T_0^i)}{\sigma})}\end{aligned}$$

where the function $\Phi(.)$ is the complementary error function and σ is its shape parameter of the lognormal distribution. For Weibull distributions we will have:

$$\begin{aligned}S(\Delta) &= 1 - \exp(\Delta^\gamma) \\ \mathbb{P}\text{rob}\{T < \tau + \Delta | T > \tau\} &= 1 - \frac{1 - \exp((\tau + \Delta - T_0^i)^\gamma)}{1 - \exp((\tau - T_0^i)^\gamma)}\end{aligned}$$

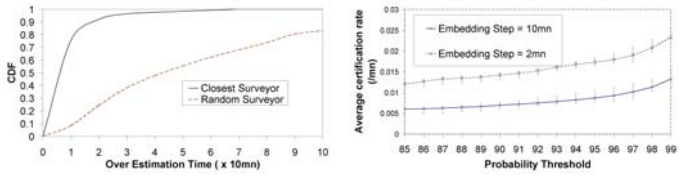
where γ is the shape parameter of the Weibull distribution.

The validity time of a certificate, Δ, is then computed by setting $\mathbb{P}\text{rob}\{T < \tau + \Delta | T > \tau\} = p_{th}$, where p_{th} is the chosen reliability of the estimate (i.e. the long-term proportions of validity periods that will expire before the corresponding coordinate change). The certificate then consists in the verified coordinate, the timestamp of the certificate creation, the validity period Δ of the certificate, as well as the identification of the node the certificate is delivered to (i.e. IP address). The certificate is then signed by the certifying Surveyor using appropriate credentials and encryption keys and issued to the node.

2) Evaluation: To evaluate the effectiveness of our certificate validity computation, we study the over-estimation resulting from each certificate: for each certificate issued, if the corresponding node moves before its current certificate expires, we record the residual time left on the certificate. This over-estimation is an important security parameter, as it is the lapse of time a malicious node could use a "stale" certificate.

Figure 6(a) shows the CDF of over-estimation times when the certificates are issued by either the closest Surveyor or a Surveyor chosen at random. We can clearly see that most certificates will not outlive their corresponding coordinates by more than 2 embedding periods in the case where they are delivered by the closest Surveyor. This is a good result, as many a time, coordinate changes in such timescale will be "localized" in space. The figure also confirms that the accuracy of the validity periods, as thus the security of the system, is improved if coordinates are certified by nearby Surveyors.

Scalability is also an important factor for any certification scheme. Although under-estimation of the validity period of certificates does not pose any security issue for the system, it does tend to increase the load on the system, and on the Surveyors in particular. Obviously, the higher the probability threshold used to compute the certificate validity time, the shorter this time will be. We therefore measure the mean validity period over all certificates for various values of the probability threshold. Figure 6(b) shows the corresponding average certification rate, which is the inverse of the mean validity period. The average certification rate gives the average number of certification requests that will be issued per node and per time unit (here the embedding period) to the system. This number, multiplied by the number of nodes in the system, and divided by the number of Surveyors and the embedding period gives a certificate request rate per second at each Surveyor. Figure 6(b) shows that the average certification rate increases gently as the probability threshold increases (i.e. as the computation becomes more conservative). This behavior shows that we can afford a reliability of 95% (and therefore high security) with moderately low overhead.



(a) CDF of Over Estimation Times (p_{th} = 0.95), Embedding period = 10mn

(b) Average Certification Rate

Fig. 6. Over estimations and average certification rates

IV. Distance Estimation Performance Evaluation

In this section, we study the impact of an attack on the accuracy of distance estimations, with and without our proposed coordinate certification defense mechanism. This attack consists for malicious nodes in choosing, once the Vivaldi system has stabilized, a target they wish to get closer to (each malicious node chooses a target at random amongst the honest nodes), and moving in a straight line towards this target by steps computed from the innovation process of their Kalman filter (see section III-A1 and [4] for details). After each displacement, each malicious node seeks to have its

newly faked coordinate certified. This strategy is designed to outsmart the tests used for coordinate verification, by only taking (fake) steps that could be attributed to normal system dynamics, as well as caused by normal noise, by the detection test. We choose such subtle attack, as more obvious displacement caused by fake coordinate would be easier to detect. Obviously, in the case where coordinate certification is employed, malicious nodes are limited to fake coordinates they can get a certificate for. We carry out this attack on our PlanetLab experimental setup, with a varying population of malicious nodes.

To assess the impact on distance estimation, we define the following metrics:

- Relative Estimation Error: $RER = \frac{|\,||C'_i - C_j|| - ||C_i - C_j||\,|}{||C_i - C_j||}$, where C_i is a node's real coordinate in the system without malicious activity, and C'_i is a node's advertised coordinate (and C'_i is either faked, certified or both).
- Security Gain Ratio: $SGR = \overline{RER}_{on}/\overline{RER}_{off}$, where $\overline{RER}_{on}$ ($\overline{RER}_{off}$ resp.) is the average RER measured between all pairs of nodes when the security mechanism is on (off resp.).

Figure 7 shows the SGR observed at the end of the experiment that was allowed to run for a considerable time after convergence of the original (clean) Vivaldi system. The curve labeled "Ratio1" depicts the SGR measured in the presence of malicious nodes, while the curve labeled "Ratio2" depicts the SGR measured in the clean Vivaldi system without malicious activity. From this figure, we can conclude that the accuracy of distance estimation in the system with malicious nodes is much improved when coordinate certification is in use than when it is not. This is because the coordinate verification phase of the certification filters out most of the faked displacements. We also see that the curve "Ratio2" exhibits a value of 1, indicating that the presence of the certification system does not degrade the performance of the clean system without malicious nodes. In other words, the coordinate certification is very much non intrusive.

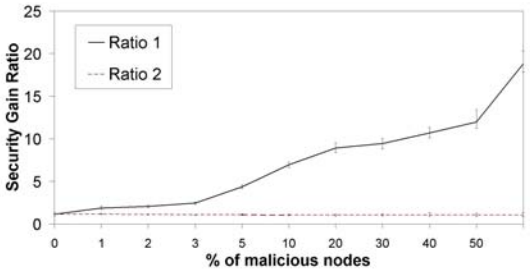


Fig. 7. Progressive Displacement Attack: Security gain ratio in function of the malicious nodes percentage in the overall population.

V. Conclusions

In this paper we have presented a coordinate certification method to protect the distance estimation phase of Internet Coordinate Systems from nodes lying about their coordinate. This work thus complements previous works on Coordinate System embedding phase security.

The proposed certification method is based on a coordinate verification method, along with a validity period estimation for the corresponding certificate. This method has been shown to be effective, exhibiting good verification test performance (high true positive detection rate with low false positive rates), while achieving a very good trade-off between scalability and security. Indeed, the validity periods of certificates are rarely over-estimated, while they still do not trigger too frequent re-certifications.

The reader should note though, that a node knows when its next embedding will occur, and thus when its coordinate is likely to change. A malicious node could exploit this knowledge to seek to obtain a certificate (for its current coordinate) just before performing this embedding, as this could leave such node in possession of a certificate for a soon to be outdated coordinate. To palliate this problem, one could envisage that Surveyors carry out "spot check" on the validity of a node's certificate: if the certified coordinate fails a new coordinate verification test, the node is "penalized" for using an outdated certificate.

Although this paper focused on Vivaldi for measurements and experimentations, the method proposed for coordinate certification is independent of the embedding protocol. This because the malicious embedding neighbor detection test that forms the basis of the coordinate verification is itself independent of the specifics of the embedding protocol, and because the validity period computation only depends on observed coordinate inter-shift times. Our proposed method would then be general enough to be applied in the context other Internet coordinate systems than Vivaldi.

References

[1] M. Costa, M. Castro, A. Rowstron, and P. Key, *Practical Internet coordinates for distance estimation*, in Proc. of the IEEE International Conference on Distributed Computing Systems, Tokyo, March 2004.
[2] T. E. Ng and H. Zhang, *A Network Positioning System for the Internet*, in Proc. of USENIX, Boston, June, 2004.
[3] F. Dabek, R. Cox, F. Kaashoek and R. Morris, *Vivaldi: A decentralized network coordinate system*, in Proc. of SIGCOMM, August, 2004.
[4] M. A. Kaafar et al., *Securing Internet Coordinates Embedding Systems*, in Proc. of SIGCOMM, Kyoto, August 2007.
[5] G. Wang et al., *Distributed Algorithms for Stable and Secure Network Coordinates*, In Proc. of IMC, October, 2008.
[6] M.Sherr et al., *Veracity: A Fully Decentralized Service for Securing Network Coordinate Systems*, In Proc. of IPTPS, February, 2008
[7] M.Sherr et al., *Towards Application-Aware Anonymous Routing*, In Proc. Hot Topics in Security (HotSec), 2007.
[8] L. Mathy et al., *Impact of Simple Cheating in Application-Level Multicast*, in Proc. of INFOCOM, San Francisco, April 2003.
[9] R. Zhang et al. *Impact of the Inaccuracy of Distance Prediction Algorithms on Internet Applications, an Analytical and Comparative Study*, in Proc. of INFOCOM, April, 2006.
[10] J. Ledlie, P. Gardner, and M. Seltzer, *Network Coordinates in the Wild*, in Proc. of NSDI, April, 2007.
[11] A. W. Moore et al., *Median Filtering for Removal of Low-Frequency Drift*, Analytic Chemistry, pp. 65:188, 1993.
[12] Y. Zhang et al., *On the Constancy of Internet Path Properties*, in Proc. of ACM SIGCOMM Internet Measurement Workshop, San Francisco, November 2001.
[13] G. R. Shorack and J. A. Wellner, *Empirical Processes With Applications to Statistics*, John Wiley&Sons Inc, 1986, pp. 976.
[14] H. Zheng et al., *Internet Routing Policies and Round-Trip Times*, in Proc. of PAM, March, 2005.

[15] E. K. Lua, et al., *On the accuracy of Embeddings for Internet Coordinate Systems*, in Proc. of IMC,October, 2005.
Proc. of SIGCOMM Internet Measurement Workshop, November 2002.