Secure Track Verification

Matthias Schäfer
University of Kaiserslautern
Germany
schaefer@cs.uni-kl.de

Vincent Lenders
armasuisse
Switzerland
vincent.lenders@armasuisse.ch

Jens Schmitt
University of Kaiserslautern
Germany
jschmitt@cs.uni-kl.de

***Abstract*—We propose a new approach for securely verifying sequences of location claims from mobile nodes. The key idea is to exploit the inherent mobility of the nodes in order to constrain the degree of freedom of an attacker when spoofing consecutive location updates along a claimed *track*. We show that in the absence of noise, our approach is able to securely verify any 2-D track with a minimum of three verifiers or any 3-D track with four verifiers. Our approach is lightweight in the sense that it considerably relaxes the system requirements compared to previous secure location verification schemes which are all agnostic to mobility. As opposed to previous schemes, our track verification solution is at the same time (i) passive, (ii) does not require any time synchronization among the verifiers, (iii) does not need to keep the location of the verifiers secret, (iv) nor does it require specialized hardware. This makes our solution particularly suitable for large-scale deployments. We have evaluated our solution in a realistic air traffic monitoring scenario using real-world data. Our results show that 25 position claims on a track are sufficient to detect spoofing attacks with a false positive rate of 1.4% and a false negative rate of 1.2%. For tracks with more than 40 claims, the false positive and false negative rates drop to zero.**

I. Introduction

The ability to track the motion of vehicles in airborne, ground, or maritime traffic control systems is a key feature to enable safe navigation and collision avoidance. For example in commercial air traffic control systems, the tracks of all aircraft are continuously monitored to inform pilots and safety personnel on the ground about potential aircraft collisions. Similarly, collision avoidance systems in future autonomous car navigation systems will require car tracking to prevent collisions at intersections [1].

A common paradigm for tracking the mobility of vehicles is to let nodes determine their own positions and broadcast them to nearby nodes. For example, in the next-generation air transportation system, aircraft determine their own positions with the aid of global navigation satellite systems such as GPS. This information is then periodically broadcast over the Automatic Dependent Surveillance - Broadcast (ADS-B) system to surrounding aircraft and sensors on the ground [2].

While this autonomous paradigm has many advantages such as relatively low costs, it makes the system vulnerable to location spoofing attacks [3], [4]. As an example related to ADS-B, Figure 1 shows the situation in which an attacker injects false position messages in order to emulate the track of a *ghost aircraft* into the air traffic surveillance and collision avoidance systems. This attack is performed by sending fake

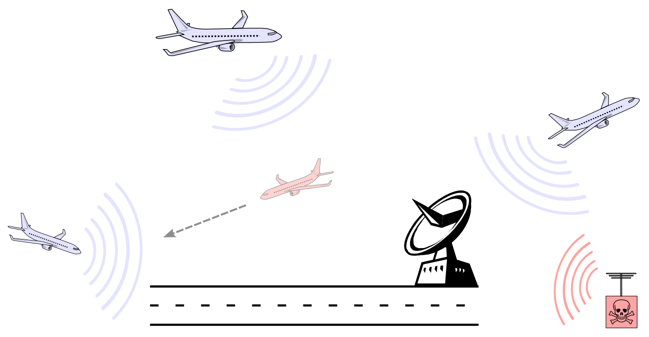
Fig. 1. Attack scenario: an attacker injects ghost aircraft to mislead the instruments of controllers and pilots. The confusion caused by such an attack can have severe or even life-threatening consequences.

position reports of airplanes which do not exist. In that way, an adversary could mislead collision avoidance systems and unmanned air vehicles or confuse air traffic controllers. It has been shown that these attacks are easy to launch on real systems [4] and the ability to verify the track claims in such systems is therefore of high importance [5].

Many schemes have been proposed in the literature to securely verify the *location* claims in wireless broadcast networks [6]–[11]. However, most of these systems are active and require specialized hardware or directional antennas. A real implementation of these protocols remains therefore often a challenge as they require expensive deployments or upgrades of existing infrastructure. Passive schemes such as multilateration [5], [12] or verification with hidden and mobile base stations [13] have also been proposed. However, passive multilateration requires a very tight time synchronization on the order of a few nanoseconds between the verifiers. Furthermore, proposals such as [13] require the verifying nodes to be at locations that are kept secret from the attacker.

In this paper, we propose a secure track verification technique which is completely passive, does not require a tight time synchronization between the verifiers, and works even if the attacker knows the positions of the verifiers. In addition, it does neither require any specialized hardware nor directional antennas. This makes the approach particularly suitable for low-cost and large-scale deployments. The core idea of our approach is to exploit the mobility of the prover to verify its position securely as it moves along a *track*. As we show in this work, the mobility of the prover is a useful dimension that can be exploited to significantly relax costly system requirements of existing location verification schemes which are agnostic to the mobility of the prover.

Our novel scheme is based on the mobility-differentiated time of arrival (ToA). In constrast to existing point-wise location verification or multilateration schemes, our approach verifies a *sequence* of location claims of a prover. We show that with this technique, it is possible to verify the track of mobile nodes locally at each receiver and, hence, avoid the need for tight synchronization among the verifiers. The verification is based on physical signal propagation constraints and local time differences at the verifier between messages that are sent by the mobile node at different positions and times.

This technique is able to correctly verify tracks from honest nodes. However, it is not secure against attacks on single verifiers. An attacker could easily adjust the transmission times of its messages to spoof any desired track for a given receiver. We thus derive the requirements for using our technique for secure verification of track claims. We show that the resulting track verification scheme is secure against attacks from a stationary adversary. The two-dimensional track of a mobile node can be constrained to a unique solution when at least three messages are received by at least three geographically distributed verifiers. With four or more distributed verifiers, any three-dimensional track can be securely verified as well.

In order to understand the performance of track verification under real-world conditions, we have further performed simulations with different levels of noise and losses at the verifiers. Additionally, we demonstrate the ability to securely verify flights as obtained from OpenSky [14], a large-scale ADS-B sensor network deployed in Central Europe. Our results suggest that our solution is able to effectively detect track spoofing attacks under realistic noise and air traffic conditions.

A. Contributions

The contributions of this work are as follows:

- We present a passive and lightweight solution to the secure track verification problem which exploits the mobility of the provers in order to relax the costly system requirements of existing solutions.
- We provide a formal analysis, proving the security of our scheme for two-dimensional and three-dimensional track verification.
- We analyze the performance of our method by conducting simulations with a realistic noise model.
- We demonstrate the feasibility to verify tracks in an air traffic monitoring scenario by using real tracks from the OpenSky sensor network.

II. Problem Statement

Similar to secure *location* verification as defined by Sastry et al. [6], we define the problem of secure *track* verification as follows: A set of verifiers V wish to check whether a prover moves on a claimed track T. A track claim consists of a sequence of location claims, that is $T = \{C_1, \dots, C_n\}$. Each location claim C_i is a tuple $(\bar{t}_i, \vec{p}_i)$, where $\bar{t}_i$ denotes a *prover-local* timestamp with its corresponding location $\vec{p}_i$. Locations are assumed to be two- or three-dimensional Euclidean coordinates. The location claim C_i is sent at time $\bar{t}_i$ and from location $\vec{p}_i$. In other words, by transmitting C_i, the prover claims to be at position $\vec{p}_i$ at the transmission time of C_i and its local, *unsynchronized* clock shows the value $\bar{t}_i$.

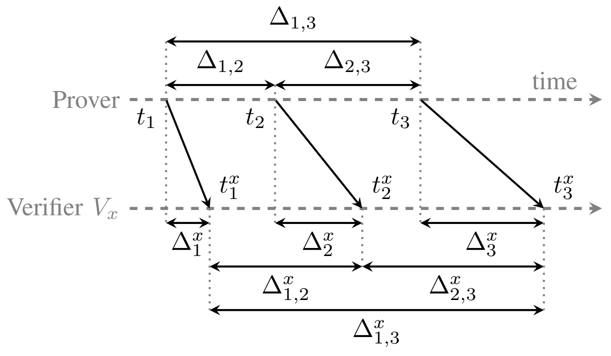

Fig. 2. The notation of time used in this paper. Timespans are denoted with an Δ, points in time with t. The indexes refer to positions and the superscripts to entities such as verifiers $(x, y, \dots)$ or an adversary (A).

Our goal is to verify tracks of moving nodes. Accordingly, we further assume that $|T| > 1$ and $p_i \neq p_j$ for at least one pair of location claims $C_i, C_j \in T$. This assumption clearly draws a distinction between our problem and that of location or in-region verification as defined in [6].

A. Time Notation

We assume that both, prover and verifiers are equipped with clocks which do not have to be synchronized – a major advantage of our scheme. Timestamps represent the local time of a node at a certain event. To distinguish between global time and local timestamps, we denote global time with t and the node-local time that corresponds to t (i.e., the timestamp at time t) with $\bar{t}$. For the theoretical analysis, we assume that all clocks run at the same speed and positions as well as timestamps are perfect, i.e., can be measured without error. The effects of clock drift and measurement error are studied in detail in Section IV.

The following temporal relationships and notations are summarized in Figure 2. The timespan between the transmissions of two location claims C_i and C_j is denoted by $\Delta_{i,j} = t_j - t_i$. The arrival time of a location claim C_i at verifier $V_x \in V$ is denoted by t_i^x. Analogously, the timespan between the arrivals of two location claims C_i and C_j at V_x is denoted by $\Delta_{i,j}^x = t_j^x - t_i^x$. The propagation delay of C_i's signal on its way to a verifier V_x is denoted by $\Delta_i^x = t_i^x - t_i$.

Values derived by local timestamps are overlined. For the following analysis, it is worth noting that for two *honest* location claims C_i and C_j, it holds that $\overline{\Delta}_{i,j} = \bar{t}_j - \bar{t}_i = \Delta_{i,j}$.

B. System Model

The system model is motivated by air traffic monitoring (ATM) systems. In the upcoming next generation air transportation system, aircraft determine their own position using satellite navigation and broadcast it periodically to surrounding ground stations. These position reports can be considered as location claims of the aircraft's track. The receiving ground stations are connected via ground networks. In aviation, facilities are usually well protected and therefore, the ground stations and the network can be assumed to be secure.

In our model, a moving prover accordingly broadcasts a sequence of location claims C_i $(i = 1, 2, \dots)$ to a set of stationary verifiers $\{V_x, V_y, \dots\}$ using a wireless communication channel. We assume that there is no compromised verifier and all verifiers are able to communicate securely with each other. Each verifier V_x knows its position $\vec{p}_x$.

Besides ATM, other systems are also well conceivable areas of application for our scheme. The key characteristic of our system model is the mobility of the prover. Therefore, any location-aware application with mobile stations (e.g., vehicular ad hoc networks or cellular networks) might be a potential target system for our scheme.

C. Adversarial Model

In order to analyze the security of our track verification scheme, we use the following threat model. We consider a single adversary A located at position $\vec{p}_A$. We assume for our theoretical analysis that it uses an omni-directional antenna to broadcast the location claims. This strong assumption ensures that all verifiers in the reception area of A receive the exact same location claims during the verification process. In section V, we propose an extension to our scheme that allows to also defend against attackers that are able to control exactly who is receiving which location claim.

The adversary has full control of the location claim's content. In particular, $\vec{p}_i$ and $\bar{t}_i$ are chosen by the attacker and the transmission time t_i of C_i does not necessarily correlate with the timestamp $\bar{t}_i$. In addition to these assumptions, the adversary also knows the exact position of all verifiers. Further adversarial models such as mobile attackers or attackers with limited knowledge are discussed in section VIII.

With respect to the ATM scenario, a realization of our threat model could be an adversary positioned close to an airport that injects fake position reports to cause confusion or prevent departures. As mentioned above, the feasibility of such attacks has been successfully demonstrated, even with low-cost hardware [3], [4].

III. Basic Verification Scheme

Using the above notations, we can conclude that for valid location claims, the inter-arrival times of the location claims C_i and C_j at verifier V_x differ from the inter-transmission times by the difference in propagation delays from $\vec{p}_i$ and $\vec{p}_j$ to $\vec{p}_x$:

$$\Delta^x_{i,j} = \Delta_{i,j} + (\Delta^x_j - \Delta^x_i) \tag{1}$$

Provers and verifiers do not have a common time base since they are not assumed to be synchronized. Yet, based on the reported and measured local timestamps, the verifier can calculate $\overline{\Delta}_{i,j}$ and $\overline{\Delta}^x_{i,j}$. The propagation delays can be estimated using $\Delta^x_{i/j} = \|\vec{p}_{i/j} - \vec{p}_x\|/c$, where c denotes the signal propagation speed and $\|\cdot\|$ the Euclidean distance.

At the core of our track verification scheme, each verifier checks for all pairs $C_i, C_j \in T$ whether the following property holds:

$$\overline{\Delta}^x_{i,j} \overset{?}{=} \overline{\Delta}_{i,j} + (\Delta^x_j - \Delta^x_i) \tag{2}$$

It is easy to see that Equation (2) is the same as Equation (1) if the prover reported its position and timestamps correctly. Thus, Equation (2) holds if the prover claimed its track honestly. Using the terminology of location verification, our scheme therefore satisfies the property of completeness [6].

Concerning the security property of our verification scheme, we claim that given a certain number of verifiers, a dishonest prover cannot send false location claims without violating Equation (2) for at least one verifier. To prove this hypothesis, we conduct a theoretical security analysis next.

A. Security Analysis

For our analysis, we assume that the adversary's goal is to claim a track with two location claims $C_1 = (\bar{t}_1, \vec{p}_1)$ and $C_2 = (\bar{t}_2, \vec{p}_2)$ with $\vec{p}_1 \neq \vec{p}_2$. We can do so without loss of generality, since Equation (2) constitutes a pairwise check for all claims in T without particular order. Hence, if our scheme is secure for arbitrary C_1 and C_2, it is also secure for track T. To provide a better understanding how security is established in our verification scheme, we analyze it step by step by increasing the number of verifiers $|V|$ which receive C_1 and C_2.

Case $|V| = 1$: With respect to the calculation done by the verifiers, we can rewrite Equation (2) as

$$\bar{t}^x_2 - \bar{t}^x_1 \overset{?}{=} (\bar{t}_2 - \bar{t}_1) + (\Delta^x_2 - \Delta^x_1) \tag{3}$$

The adversary's goal is to find a transmission time t_2 for C_2 relative to t_1, such that the inter-arrival time at V_x (left-hand side of the equation) corresponds to the location claims C_1 and C_2 (right-hand side). Therefore, it waits for the appropriate time Δ_A after the transmission of C_1, i.e. $t_2 = t_1 + \Delta_A$. By plugging this into Equation (3), we obtain

$$\underbrace{(\overbrace{t_1 + \Delta_A}^{=t_2} + \Delta^x_A)}_{=t^x_2} - \underbrace{(t_1 + \Delta^x_A)}_{=t^x_1} = (\bar{t}_2 - \bar{t}_1) + (\Delta^x_2 - \Delta^x_1)$$

with $\Delta^x_A = \|\vec{p}_A - \vec{p}_x\|/c$ being the propagation delay from the adversary to V_x. This equation can be solved for Δ_A and thus, the attacker can simply use

$$\Delta_A \quad = \quad (\bar{t}_2 - \bar{t}_1) + (\Delta^x_2 - \Delta^x_1)$$

to spoof C_1 and C_2.

This result means that adversaries are able to spoof arbitrary tracks by simply adjusting the time between the transmissions of the location claims if there is only one verifier (and they are close enough to the verifier). An illustration of such an attack is provided in Figure 3.

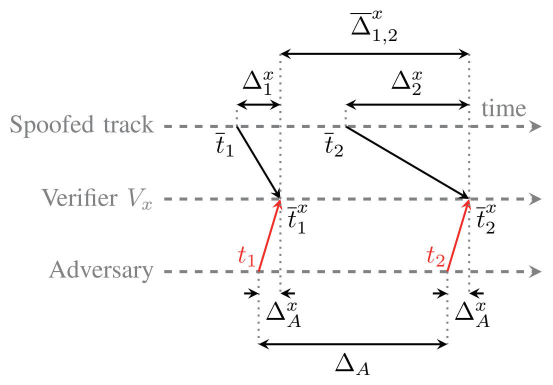

Fig. 3. Illustration of an attack on a single verifier. By adjusting the transmission time t_2 with respect to t_1 and the desired $\overline{\Delta}^x_{i,j}$, the adversary's location claims C_1 and C_2 are received by the verifier V_x as they were sent from the spoofed track.

Case $|V| = 2$**:** As above, the adversary tries to forge a location claim by adapting the difference in transmission times $\Delta_{1,2}$ such that its location claims C_1 and C_2 seem honest for two verifiers V_x and V_y. The difference to the previous case is that the adversary's signal now experiences two independent propagation delays Δ^x_A and Δ^y_A and both verifiers expect independent propagation delays Δ^x_i and Δ^y_i. As a result and similar to the previous case, the adversary has to find Δ_A such that the following system of equations is satisfied:

$$\begin{aligned}\Delta_A &= (\bar{t}_2 - \bar{t}_1) + (\Delta^x_2 - \Delta^x_1)\\ \Delta_A &= (\bar{t}_2 - \bar{t}_1) + (\Delta^y_2 - \Delta^y_1)\end{aligned}$$

Hence, the adversary is limited in its choice of $\vec{p}_1$ and $\vec{p}_2$. In particular, it has to choose them such that

$$\Delta^x_2 - \Delta^x_1 = \Delta^y_2 - \Delta^y_1.$$

Without loss of generality, we fix position $\vec{p}_1$ for the first location claim. Then, the adversary can use any position $\vec{p}_2 \in H(\vec{p}_1, \vec{p}_x, \vec{p}_y)$ for its second location claim C_2 with

$$H(\vec{p}_1, \vec{p}_x, \vec{p}_y) = \{\vec{p} \in \mathbb{R}^n \mid \|\vec{p} - \vec{p}_x\| - \|\vec{p} - \vec{p}_y\| = \|\vec{p}_1 - \vec{p}_x\| - \|\vec{p}_1 - \vec{p}_y\| \}$$

where n is the number of dimensions. In the two-dimensional case, this set of positions $H(\vec{p}_1, \vec{p}_x, \vec{p}_y)$ corresponds to one arm of a hyperbola with foci $\vec{p}_x$ and $\vec{p}_y$ and a difference of distances to the foci of $\|\vec{p}_1 - \vec{p}_x\| - \|\vec{p}_1 - \vec{p}_y\|$. With $n = 3$, H is one sheet of a hyperboloid with the same parameters.

The key insight is that the adversary cannot claim arbitrary tracks anymore. In particular, it loses one degree of freedom with the introduction of a second verifier. It is limited in its choice for the second position $\vec{p}_2$ to positions that lie on $H(\vec{p}_1, \vec{p}_x, \vec{p}_y)$.

In conclusion, the adversary can still spoof tracks that go through one arbitrary position of interest. Although this might be sufficient for some attacks, being restricted to a hyperbola is already a significant limitation. Furthermore, the two verifiers can easily check whether the locations of the track lie on such a hyperbola. In case they do, they can consider the track being suspicious. In scenarios where hyperbolic tracks are impossible (e.g. roads in a vehicular network), attacks would not remain undetected.

Case $|V| = 3$**:** Analogously to the previous case, we can derive the constraint

$$\Delta^x_2 - \Delta^x_1 = \Delta^y_2 - \Delta^y_1 = \Delta^z_2 - \Delta^z_1$$

for two location claims C_1 and C_2 and three verifiers V_x, V_y, and V_z. This constraint can only be satisfied by an adversary if it forges the location claims C_1 and C_2 such that the pairwise hyperbolas (or hyperboloids, respectively) with the three verifiers intersect at $\vec{p}_1$ and $\vec{p}_2$. That means, for a position $\vec{p}_1$, $\vec{p}_2$ it must satisfy the following constraint:

$$\bigwedge_{\substack{\{V_x, V_y\} \in V \\ V_x \neq V_y}} \vec{p}_2 \in H(\vec{p}_1, \vec{p}_x, \vec{p}_y) \tag{4}$$

We now analyze these intersections. For the sake of concise presentation, we only consider the two-dimensional case. Extending our results to three dimensions is straightforward: intersections of hyperboloids instead of hyperbolas must be considered.

With $d_{xy}(\vec{p}_1) = \|\vec{p}_1 - \vec{p}_x\| - \|\vec{p}_1 - \vec{p}_y\|$ and $\vec{p}_i = (x_i, y_i)$, we can set up a system of equations for the intersections of two hyperbolas $H(\vec{p}_1, \vec{p}_x, \vec{p}_y)$ and $H(\vec{p}_1, \vec{p}_x, \vec{p}_z)$. Each intersection $(x, y) \in \mathbb{R}^2$ must be a solution for the system of equations derived by the definition of H:

$$\sqrt{(x - x_x)^2 + (y - y_x)^2} - \sqrt{(x - x_y)^2 + (y - y_y)^2} = d_{xy}(\vec{p}_1)$$
$$\sqrt{(x - x_x)^2 + (y - y_x)^2} - \sqrt{(x - x_z)^2 + (y - y_z)^2} = d_{xz}(\vec{p}_1)$$

By squaring and rearranging these equations, we obtain

$$\sqrt{(x - x_x)^2 + (y - y_x)^2} = x \cdot c_1 + y \cdot c_2 + c_3 \tag{5}$$
$$\sqrt{(x - x_x)^2 + (y - y_x)^2} = x \cdot c_4 + y \cdot c_5 + c_6 \tag{6}$$

with constants

$$\begin{aligned}c_1 &= (x_x - x_y)/d_{xy}(\vec{p}_1)\\ c_2 &= (y_x - y_y)/d_{xy}(\vec{p}_1)\\ c_3 &= (x_y^2 + y_y^2 - x_x^2 - y_x^2 + d_{xy}(\vec{p}_1)^2)/(2d_{xy}(\vec{p}_1))\\ c_4 &= (x_x - x_z)/d_{xz}(\vec{p}_1)\\ c_5 &= (y_x - y_z)/d_{xz}(\vec{p}_1)\\ c_6 &= (x_z^2 + y_z^2 - x_x^2 - y_x^2 + d_{xz}(\vec{p}_1)^2)/(2d_{xz}(\vec{p}_1))\end{aligned}$$

Subtracting Equation (6) from Equation (5) results in

$$y = x \cdot \frac{c_1 - c_4}{c_5 - c_2} + \frac{c_3 - c_6}{c_5 - c_2}$$

Plugging this equation into one of the initial equations results in a quadratic equation for x and y. Quadratic equations have either zero, one, or two solutions. In our case, we even know that by construction of the hyperbolas, it has at least one solution, that is $\vec{p}_1$. Thus, there is either no or at most one possible position left for the adversary to spoof a track without violating Equation (2) for one of the verifiers.

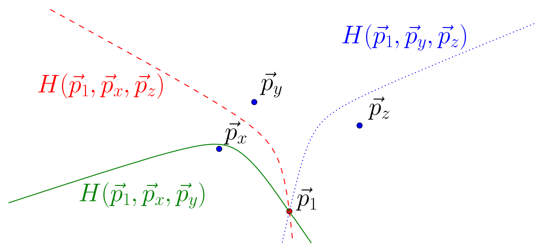
$H(\vec{p}_1, \vec{p}_y, \vec{p}_z)$
$H(\vec{p}_1, \vec{p}_x, \vec{p}_z)$
$\vec{p}_y$
$\vec{p}_z$
$\vec{p}_x$
$H(\vec{p}_1, \vec{p}_x, \vec{p}_y)$
$\vec{p}_1$

Fig. 4. Example with three verifiers and their pairwise hyperbolas for a claimed position $\vec{p}_1$. No further position can be spoofed without being detected since there is no other intersection.

A scenario with three verifiers and their pairwise hyperbolas is depicted in Figure 4. The adversary wants to spoof a track and claims to be at $\vec{p}_1$. As there is no further intersection of the three hyperbolas, it cannot claim any second position without being detected by at least one verifier.

In three dimensions and similar to navigation methods based on time-difference of arrival measurements (e.g. multilateration), a fourth verifier would be necessary to pin the attacker down to a single position. In general, $|V|$ verifiers result in $|V| - 1$ pairwise hyperboloids. With three-dimensional locations and $|V| = 3$, the two hyperboloids intersect on a curve. As in the two-dimensional case, adding a fourth verifier reduces the number of intersections to at most two points in space.

Case $|V| > 3$: Equation (4) is a general result which also holds for more than three verifiers. For the two-dimensional case, the guarantees given by three verifiers are already sufficient since attacks using tracks with two intersections can simply be prevented by requiring $|T| \geq 3$. However, more than three verifiers can be beneficial to mitigate noise in the verification data such as measurement errors or clock drifts. This interesting issue of imperfect verification data and how to use $|V| > 3$ to improve the accuracy of verification is dealt with below in Sections IV and VI.

B. Conclusions from the Analysis

The above analysis shows that the adversary loses one degree of freedom with each additional verifier. The intuition behind this is as follows: As the adversary is changing its position between individual location claims, the propagation delay to each verifier must also change in order to satisfy Equation (2) at all verifiers. Thus, adversaries would have to vary the propagation delays to each of the verifiers independently to successfully pretend movement. Since all verifiers receive the same messages (due to the broadcast transmission), this is not possible. As a result, the only spoofable track for a stationary adversary is the track on which the difference in propagation delay to each verifier is constant. For two verifiers, this is a hyperbola. For more than two verifiers, this property only holds for the intersections of the pairwise hyperbolas (see Figure 4).

In summary, we can generalize the assumptions for secure track verification as follows. Let n be the number of dimensions, that is $\vec{p}_i \in \mathbb{R}^n$. Then our scheme can detect track spoofing attacks if any of the following two conditions is met:

1) $|V| \geq n \wedge \exists_{\vec{p}_1,\vec{p}_2 \in T}\ p_2 \notin H(\vec{p}_1, \vec{p}_x, \vec{p}_y)$: The location claims of T are received by at least n verifiers and there are two different positions in T, where one position does not lie on the hyperbola (or the hyperboloid, respectively) spanned by the other position and the two verifiers' positions.

2) $|T| \geq 3 \wedge |V| \geq n+1$: The track consists of at least three location claims and the claims are received by at least $n+1$ verifiers.

IV. Dealing With Noise

In practice, verifiers have to deal with imperfect verification data since time and position measurements are error-prone. For instance, clocks have different speeds which results in non-negligible drifts. In order to assess the practicality and performance of our verification scheme under realistic conditions, we use the following error model.

A. Error Model

1) Clock Drift: The speed of clocks is highly dependent on environmental conditions such as pressure or temperature [15]. However, we assume that the duration of the verification process is on the order of seconds or minutes. Most environments (such as the interior of vehicles) are sufficiently stable within such time periods. Hence, we assume that clock drift is linear and thus increases at a constant rate during the verification process.

In accordance to that, we model clock drift as follows. The error due to clock drift ϵ_{drift} linearly depends on the duration between two time measurements. It can be modeled by a drift coefficient t_{drift} for an entity X. Assuming that X wants to measure a period of time $\Delta_{i,j} = t_j - t_i$, the clock drift error ϵ_{drift} of X's measurement $\overline{\Delta}_{i,j}^{X}$ is given by

$$\epsilon_{drift} = \Delta_{i,j} \cdot t_{drift} = (t_j - t_i) \cdot t_{drift}.$$

2) Measurement & Channel Noise: Measuring points in time at which events occur always involves measurement errors. For instance, systems are clocked by an oscillator at a certain rate and they only perform actions if a pulse or pulse edge of the oscillator is present. Hence, observations can only be made at discrete points in time. This leads to measurement errors when events of interest (such as the arrival of a signal) occur between two clock ticks. Besides timing errors, wireless transmission characteristics such as multipath propagation distort the signal. This may also results in noise when determining timestamps for signal arrivals. In addition to erroneous time measurements, our scheme may also suffer from erroneous position information. If provers use GPS to determine their positions, the location claims may contain errors of up to 15 m.

We assume that measurement and channel noise are independent for each location claim and each of its associated timestamps. In accordance with [16], we summarize all sources of noise in a zero-mean Gaussian random variable $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The variance σ^2 depends on the accuracy of the system components involved in the verification process. For instance, if clocks with higher rates are used, σ^2 becomes smaller.

By combining clock drift and noise, we conclude that the error contained in measuring a time period $\Delta_{i,j}$ for our verification scheme can be modeled as

$$\overline{\Delta}_{i,j} - \Delta_{i,j} = \epsilon_{drift} + \epsilon = (t_j - t_i) \cdot t_{drift} + \epsilon \qquad (7)$$

In the following, we propose two versions of our scheme: local and global verification. In local verification, verifiers calculate and check their verification results locally. They do not need to communicate with each other. This has the advantage, that communication overhead is minimal and verifiers do not have to be connected. They could simply send an alarm to a central entity in case an attack was detected. This simplicity, however, comes at a price. Local verification does not take full advantage of the total number of verifiers. Therefore, we also propose a global scheme, which is based on the local scheme but verifiers collaborate in order to reduce the impact of noise. We conclude the section with a comparison of both approaches.

B. Local Track Verification Scheme

The noise in real systems makes a simple check of Equation (2) to verify a track impractical. Therefore, we adapt our basic verification scheme to deal with noisy values. The idea is to use all received location claims to estimate the error. As shown in [16], jointly estimating clock drift and measurement error is not feasible since the Cramer-Rao lower bound of the estimation error is too large. Therefore, we perform our verification in two steps.

The first step estimates the clock drift. This estimate is then used in the second step to cancel out ϵ_{drift} from our measurements. Let $n = |T|$ be the number of received location claims. The clock drift coefficient t^x_{drift} of V_x can then be estimated using

$$\hat{t}^x_{drift} = \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{\overline{\Delta}^x_{i,j}}{\Delta_{i,j} + (\Delta^x_j - \Delta^x_i)} - 1 \right) \qquad (8)$$

From a security perspective, estimating the clock drift in this way raises the question whether an adversary can take advantage of pretending certain clock drifts or not. The answer to that question is no. Since we do not make any assumptions on clock drifts, a fake clock drift is just as good as a true one, and both will be equally eliminated by Equation (8). Faking different clock drifts during one track is even worse, since the estimation error will be high and, thus, increase the final verification result $\mathcal{V}^x_T$ defined below (which leads to a rejection of the claimed track). Hence, fake clock drifts do not pose a threat to our scheme.

The verification is finally done in the second step by calculating the mean squared error when subtracting the right-hand side from the corrected left-hand side of Equation (2). We denote this *local verification result* $\mathcal{V}^x_T$ of Verifier V_x for a track T by

$$\mathcal{V}^x_T = \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left((\Delta_{i,j} + (\Delta^x_j - \Delta^x_i)) \cdot (1 + \hat{t}^x_{drift}) - \overline{\Delta}^x_{i,j} \right)^2 \qquad (9)$$

The results of our security analysis in Section III-A imply that for honest track claims, $\mathcal{V}^x_T$ should converge to the average squared error. For dishonest claims, $\mathcal{V}^x_T$ must be higher for at least one verifier due to the deviation caused by its dishonesty. In our track verification scheme, each verifier $V_x \in V$ calculates $\mathcal{V}^x_T$ and checks whether it is below a predefined threshold. In case a verifier's local result is higher than the threshold, the verification fails and the track is considered to be dishonest. We call this verification process *local track verification* as each verifier calculates its verification result locally. Accordingly, the threshold for the local verification result is denoted by $\mathcal{T}_{local}$.

The threshold for the local verification should be chosen based on the variance σ^2 of the measurement error ϵ and the number of location claims n. As n increases, $\hat{t}^x_{drift}$ becomes more accurate and $\mathcal{V}^x_T$ is supposed to converge to a value close to zero. An optimal $\mathcal{T}_{local}$ must fulfill the same properties as a location verification scheme according to [6]:

1) Completeness: If T is an honest track claim, $\mathcal{V}^x_T < \mathcal{T}_{local}$ must hold for all verifiers $V_x \in V$.

2) Security: If T is a false track claim, $\mathcal{V}^x_T \geq \mathcal{T}_{local}$ must hold for at least one verifier if one of the constraints given in Section III-B holds.

If such an optimal threshold exists, the local verification scheme is able to perfectly distinguish between honest and dishonest track claims. For later analyses and optimizations, we measure the "optimality" of $\mathcal{T}_{local}$ and our system in terms of *false rejection* and *false acceptance rates*. A false rejection of a track means the detection of an attack, although the prover is honest. A false acceptance occurs if a false track claim is not rejected by the system. Both rates can be controlled with $\mathcal{T}_{local}$. On the one hand, if $\mathcal{T}_{local}$ is smaller than the highest possible $\mathcal{V}^x_T$ for honest tracks, false rejections can occur. On the other hand, false acceptances are possible if $\mathcal{T}_{local}$ is greater than the smallest possible $\mathcal{V}^x_T$ for false track claims.

C. Global Track Verification Scheme

As our evaluation below shows, bad hardware accuracy and verifier placement may result in false rejections and acceptances by our local verification scheme. The local verification scheme, however, does not take advantage of the total number of verifiers since it only considers local results. Given a higher number of verifiers, a better verification decision can be made by obtaining all local results instead of considering them separately. We call this extension *global track verification*.

In our global verification scheme, the verifiers exchange their verification results $\mathcal{V}^x_T$ and each verifier calculates the average verification result:

$$\mathcal{V}_T = \frac{1}{|V|} \cdot \sum_{V_x \in V} \mathcal{V}^x_T \qquad (10)$$

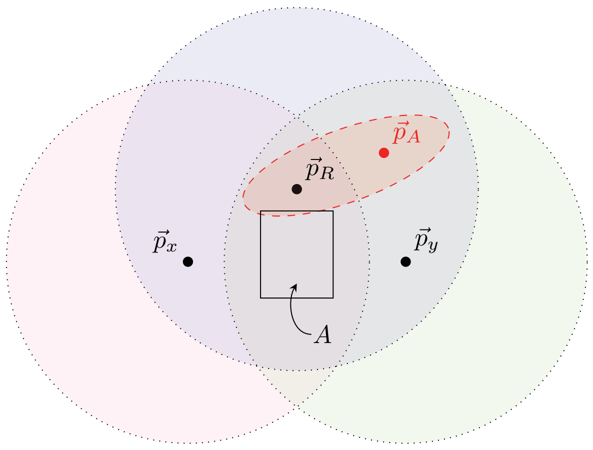

Fig. 5. Example scenario: three verifiers V_x, V_y, and R (located at $\vec{p}_x$, $\vec{p}_y$, $\vec{p}_R$) are arranged such that the area of interest for R is covered by the three verifiers reception ranges (dotted circles). The attacker (located at $\vec{p}_A$) transmits its signal to R using a directed antenna (dashed area) and avoids being detected by the other verifiers.

Similar to $\mathcal{T}_{local}$, we can define a threshold $\mathcal{T}_{global}$ for $\mathcal{V}_T$. A track T is then accepted by the system if $\mathcal{V}_T < \mathcal{T}_{global}$ and rejected if $\mathcal{V}_T \geq \mathcal{T}_{global}$ holds.

The choice, whether to use the local or the global verification scheme depends on hardware constraints and infrastructure. In case it is cheaper to distribute many low-cost verifiers instead of a few high-end devices, the global verification is preferable. If verifiers are equipped with very accurate hardware, the local check might be the better choice as it is more sensitive to anomalies. Besides that, the local verification scheme produces less communication overhead and does not require a fully connected network of verifiers.

V. Reception Area Sanity Check

As discussed in Section II-C, we assume that the adversary uses an omni-directional antenna. In this section, we discuss a simple extension to our scheme to basically "catch" an adversary that does not conform to this assumption. The alert reader might already have noticed the problem related to an attacker not using an omni-directional antenna: Assume receiver R wants to monitor tracks in a certain area of interest A. To securely verify the claimed tracks in A, there are $v \geq 3$ verifiers deployed such that their reception ranges cover A. R itself is one of the verifiers.

Remember the practical example from the introduction where a ground station receives tracks from aircraft and provides them to air traffic controllers who are responsible for managing the traffic in A. Now, an adversary wants to inject false tracks in order to mislead the controllers. It could transmit its track claim with a directional antenna such that it is only received by R. As R would be the only receiver, the attacker could adjust its transmission times as shown in Section III-A (case $|V| = 1$) and the attack would not be detected. This scenario is depicted in Figure 5.

This attack is possible because our verification scheme implicitly assumes that location claims are always received by all verifiers (in the transmission area of an omnidirectional prover). To address this problem, we extend our scheme with the following protocol.

A. Sanity Check Protocol

We assume that each verifier knows its reception area and can check whether a position $\vec{p}$ lies within the reception range using an indicator function $R_x(\vec{p})$:

$$R_x(\vec{p}) = \begin{cases} 1 & \text{if } \vec{p} \text{ lies within } V_x\text{'s reception range} \\ 0 & \text{else} \end{cases} \tag{11}$$

In an obstacle-free line-of-sight communication scenario, where the communication is only limited by the free-space path loss, R_x would be

$$R_x(\vec{p}) = \begin{cases} 1 & \text{if } \|\vec{p}_x - \vec{p}\| \leq r \\ 0 & \text{else} \end{cases}$$

for the maximum reception range r. For more complex reception areas, an initial sampling phase or a more sophisticated propagation model can be used to determine R_x.

In principle, the following algorithm simply checks whether the reception of the location claim C_i is normal or not. If the location claim was received although the position is not in its reception range, something is suspicious and an alarm is raised. In case the reception range covers $\vec{p}_i$, the location claim is accepted and the other verifiers are notified about the reception. For each reception of a location claim $C_i = (\bar{t}_i, \vec{p}_i)$, verifier V_x performs the following verification procedure:

```
if $R_x(\vec{p}_i) = 0$ then // I shouldn't have received this claim
  broadcastAlert($C_i$) // alert all verifiers
else
  if $C_i \notin N$ then // I received it first
    broadcastNotification($C_i$) // notify all verifiers
  end if
  $T = T \cup \{C_i\}$ // add claim to track
end if
```

where N is the set of all received notifications.

The second part of the protocol checks whether location claims are always received by all verifiers which cover $\vec{p}_i$. Assuming that the notification was sent by the verifier with the shortest distance to the prover, all other verifiers should receive the claim at latest after the difference in propagation delays. Let $\epsilon_{\max}$ be an upper bound for the maximum expected measurement error. For each received reception notification $N_i = (C_i)$ from V_y, verifier V_x performs the following procedure:

```
if $R_x(\vec{p}_i) = 1$ then // I should also receive this claim
  wait($\Delta_i^x - \Delta_i^y + \epsilon_{\max}$) // wait for it
  if $C_i \notin T$ then // I should have received it by now
    broadcastAlert($C_i$) // alert all verifiers
  end if
else
  $N = N \cup \{C_i\}$ // save notification
end if
```

The protocol raises an alarm in two cases. First, a verifier receives a location claim from a position which is under legitimate conditions not within its reception range. In this case, the prover must either be at a position other than the claimed one or use anomalous parameters such as a higher transmit power. Second, a verifier does not receive a claim it should receive under normal conditions. Both cases indicate a false track claim and attacks such as the one above are detected.

B. Channel Loss

Message loss is a natural phenomenon in wireless channels. In ADS-B for instance, message loss rates of up to 40% in peak traffic periods have been reported [4]. However, channel loss results in alarms not caused by an adversary. Thus, a single alarm does not necessarily indicate an attack and the sanity check must tolerate some loss in practice. We propose a simple statistical scheme here to do so.

Let $m = |T|$ be the number of transmitted messages, $v = |V|$ the number of verifiers, and n_a the number of alarms due to channel loss. Under the assumption that loss is a Bernoulli process[1] and the loss probability p is known, the expected number of alarms due to channel loss is $E(n_a) = v \cdot m \cdot p$. Furthermore, the number of alarms due to channel loss, n_a, is binomially distributed and we can easily build a confidence interval for n_a. Thus, a track T only passes our sanity check successfully, if n_a is within this confidence interval for a given confidence level α. In other words, if T passes the sanity check, we can be certain with a confidence of α, that the alarms are caused by channel loss, otherwise we detect an attack.

C. Security

Using the confidence interval check has certain advantages. By choosing an appropriate confidence level, a user can control the false positive and false negative detection rate. For instance, higher confidence levels result in wider confidence intervals. This, on the one hand, offers the adversary a wider scope for its attacks but, on the other hand, the false rejection rate for legitimate tracks will be decreased. In practice, a trade-off has to be found for the concrete application scenario.

For applications where provers report their track over longer periods, another advantage is the behavior of the confidence interval if m increases. In particular, the confidence interval becomes smaller for each additional location claim at an exponential rate. Figuratively speaking and in terms of security, the sanity check tightens the noose on the attacker with each additional location claim.

Most importantly, by employing this sanity check along with our track verification scheme, we force adversaries to send their location claims to all verifiers which cover the spoofed positions. This requirement issues a big challenge for realistic attackers. In order to launch an attack, they have to know the exact reception ranges of all verifiers and have to be able to control exactly which verifiers receive which location claims. In addition to that, they have to make sure, that the channel loss of their claims is similar to that of honest provers.

[1]more complex loss models or a sampling phase can be used analogously

TABLE I
OVERVIEW ON THE SIMULATION PARAMETERS FOR THE ERROR PROPAGATION ANALYSIS.

Parameter	Description
r	The radius of the circular area around the verifier
m	The number of messages per track, i.e. $m = \|T\|$
v	The number of verifiers that receive the provers location claims, i.e. $v = \|V\|$
σ	The standard deviation of the measurement error
σ_{drift}	The standard deviation of the random clock drift coefficient t^x_{drift} of the verifiers
Constant	**Description/Value**
c	The propagation speed of the signal is fixed to the speed of light (299792458 m/s)

VI. Error Propagation Analysis

This section provides insights on the requirements, performance, and security of our approach. We conducted extensive simulations and analyzed the effect of measurement error, clock drift, and number of claims on the verification result. To draw conclusions on the security (i.e. on false rejection and false acceptance rates), we compare the verification results of honest and dishonest track claims.

In order to keep the detection time low, it is desirable to keep the number of required messages as small as possible. Therefore we assume that the drift estimator $\hat{t}^x_{drift}$ is calculated with the same set of claims as the verification value $\mathcal{V}^x_T$. As a result, they are not independent and since $\hat{t}^x_{drift}$ is used to calculate $\mathcal{V}^x_T$, the error propagation in our scheme is complex and hard to analyze formally. While we know that the variance of $\hat{t}^x_{drift}$ can be estimated with

$$Var(\hat{t}^x_{drift}) = \frac{\sigma^2}{\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\left(\Delta_{i,j} + (\Delta^x_j - \Delta^x_i)\right)^2 \cdot \binom{m}{2}^2}$$

and the average estimation error converges to zero with increasing m, we cannot set up a trivial error model for $\mathcal{V}^x_T$ analogously. However, to analyze the error propagation nevertheless, we implemented the local and global verification schemes as a discrete-event simulation.

A. Simulation Setup

Initially, we assigned a random clock drift t^x_{drift} to each verifier V_x. We draw t^x_{drift} from a zero-mean Gaussian distribution with standard deviation σ_{drift}. The signal propagation speed is fixed to the speed of light (299792458 m/s) for all simulations. To cancel effects caused by tracks with special properties[2], the prover moves on random tracks for this analysis. Real tracks are considered below in Section VII. The location claims for each track are randomly chosen from a circular area A with radius r around the verifier's position. The prover's maximal change in distance to the verifier (and thus the

[2]e.g. errors due to a bad dilution of precision

mobility-differentiated time of arrival which is considered by our verification scheme) is limited by r. The unit for distances is meters. Times and periods are in seconds. Our simulation parameters and constants are summarized in Table I.

A result of our formal security analysis in Section III-A is that an implementation of our scheme must always ensure that an area of interest is always covered by at least three verifiers. If this is the case, we know that for at least one verifier V_x, the period $\overline{\Delta}_{i,j}^{x}$ differs from the expected inter-arrival time (Equation (2)). To produce valid insights on the security of our scheme, we are particularly interested in $\mathcal{V}_T^x$ of this verifier. Without loss of generality, we consider only one verifier in each simulation run and assume that it is the one we are interested in, namely the one for which Equation (2) does not hold. We generate the deviation of the adversary's signal arrival times from those of the honest prover by simply putting the adversary at a random but fixed position within A. The adversary then claims the same track as the honest prover and uses the same transmission times. The magnitude of the deviations can be controlled by r (larger r leads to larger deviations).

B. Clock Drift & Measurement Error

To keep our simulations realistic, we had to find appropriate parameters for our error model. With regard to our later analysis of our scheme's performance in a realistic air traffic scenario, we choose σ and σ_{drift} based on experiences from the OpenSky Network[3]. The OpenSky Network is a low-cost sensor network which monitors air traffic at a large scale and provides the data to researchers [14]. Low-cost receivers are distributed to volunteers in Europe and these collect (among other things) the position reports periodically broadcast by aircraft. The receivers used by OpenSky provide timestamps with a 50 ns precision for the arrival of position reports. Besides that, most aircraft are using GPS to determine their positions. The typical position accuracy of GPS is about 15 m. That leads to an estimation error of propagation delays Δ_i^x of about 50 ns. Therefore, choosing $\sigma = 50$ ns for the measurement error seems appropriate.

It is worth mentioning here, that this is a rather pessimistic assumption. The OpenSky network is using low-cost receivers which are not equipped with particularly good clocks. For better devices, timestamps with higher precision can be assumed. Furthermore, the Federal Aviation Administration (FAA) in the US is implementing navigation systems for civil aviation which can reduce positioning error to less than a meter [17].

To determine an appropriate standard deviation for clock drift errors, we considered the drift of OpenSky's receivers relatively to each other. To determine the clock drifts of the sensors, we used position reports received by multiple stations. By subtracting the difference in propagation delays to each receiver from the reception timestamps, we were able to obtain the offsets of the clocks over time and thus, the clock drift. We observed the clock drifts of eight receivers over a period of one hour and we found that they were constantly linear during that period. At this point it must be mentioned that the receivers are mostly indoors and not exposed to extreme temperature variation. The results are shown in Figure 6. According to these results, we choose a pessimistic standard deviation of 20 µs per second for the clock drifts of the verifiers σ_{drift}.

[3]http://opensky-network.org

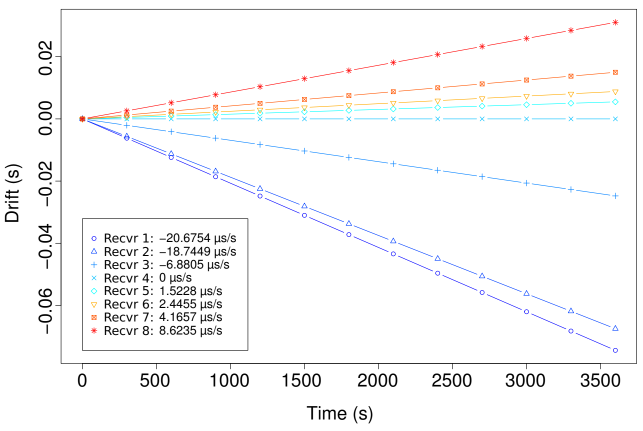

Fig. 6. Estimated clock drifts of 8 receivers of the OpenSky network over one hour. The drifts are relative to the clock of receiver 4.

C. Simulation Results

1) **Local Verification Scheme:** We first look at the local verification scheme as it is the basis for the global scheme. The goal of this analysis is twofold. On the one hand, we want to determine the least number of location claims needed to verify a track under the above error model. On the other hand, we are also interested in the benefits of receiving more location claims than actually needed. Ideally, the difference in $\mathcal{V}_T^x$ between honest and dishonest tracks becomes more pronounced with each additional location claim as the estimators of our scheme become more accurate.

To draw inferences from the local simulation results about the overall performance of our verification scheme, we compare the maximum $\mathcal{V}_T^x$ of 1000 honest tracks with the minimum $\mathcal{V}_T^x$ of 1000 dishonest tracks. In doing so, we check whether the worst verification result of the honest tracks is greater than the best verification result of the dishonest tracks. If this is the case, we can conclude that $\mathcal{T}_{local}$ does not exist since we cannot perfectly distinguish honest from dishonest tracks. Let $\max_{honest}$ be the maximum verification result for the honest tracks and $\min_{dishonest}$ the minimum verification result for dishonest tracks. We then use the "best-evil-to-worst-good ratio"

$$EGR := \min_{dishonest} / \max_{honest}$$

as response variable for our simulations. This ratio can be interpreted as follows. If the $EGR \leq 1$, $\mathcal{T}_{local}$ does not exist. Otherwise, there is a *secure interval*

$$\Sigma = (\max_{honest}, \max_{honest} \cdot EGR)$$

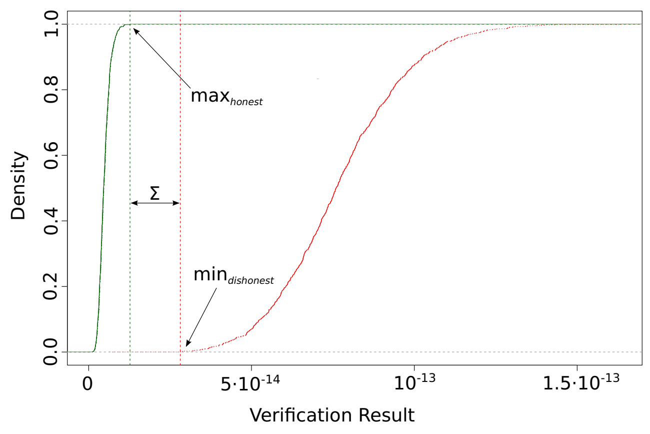

Fig. 7. Empirical cumulative distribution functions of the verification results for honest (left curve) and dishonest (right curve) tracks with $r = 200$ m, $\sigma = 50$ ns, $\sigma_{drift} = 20$ µs/s, and $m = 19$. Any $\mathcal{T}_{local}$ between max$_{honest}$ and min$_{dishonest}$ (i.e. $\mathcal{T}_{local} \in \Sigma$) perfectly separates honest and dishonest tracks.

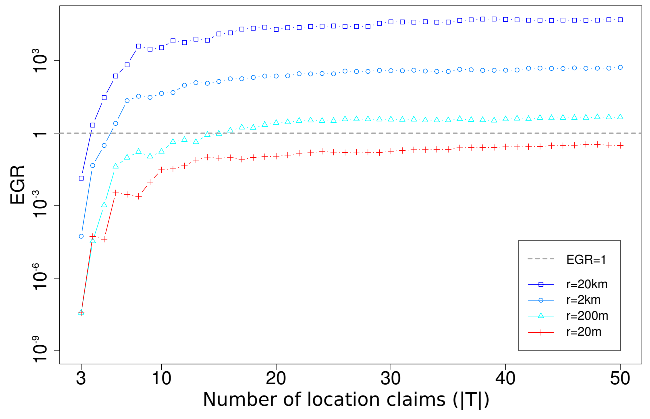

Fig. 8. Result of our simulation for different radii and increasing m. The "best evil-to-worst good ratio" (EGR) is defined as the ratio of the best dishonest result divided by the worst honest result. If $EGR > 1$ holds, there exists an optimal threshold $\mathcal{T}_{local}$ which perfectly separates honest from dishonest tracks.

where any $\mathcal{T}_{local} \in \Sigma$ results in zero false rejections and zero false acceptances for the 1000 simulated tracks and the given configuration.

Figure 7 shows the empirical cumulative distribution function of the local verification results of an example simulation. In this particular example, we used a radius of 200 m and 19 messages per track. The *EGR* is 2.854879 which is greater than 1 and hence, $\mathcal{T}_{local} \in \Sigma$ exists. In fact, the estimated secure interval for this configuration is

$$\Sigma = (1.147019 \cdot 10^{-14}, 3.274601 \cdot 10^{-14})$$

As mentioned above, the verification result is directly dependent on the simulation radius r. Therefore, we repeated our simulations for different radii. Transferred into a real-world scenario, a greater radius means larger distances between location claims. The results of the simulations are shown in Figure 8. For the radii 200 m, 2 km, and 20 km, the EGR becomes greater than 1 after a few location claims. In fact, for radii on the order of kilometers, dishonest tracks are perfectly distinguishable from honest tracks after 4 location claims.

If r becomes too small, $\mathcal{T}_{local}$ does not exist anymore. For instance, the maximum EGR for $r = 20$ m is 0.34. That means that there is no optimal threshold $\mathcal{T}_{local}$ which perfectly separates honest from dishonest tracks. This result, however, is natural since we have chosen a standard deviation for the measurement error which does not enable us to measure such small changes in propagation delay. For $r = 20$ m, the change in distance (and thus propagation delay) for random tracks is on average 10 m, but we have chosen a standard deviation of 15 m for the measurement error.

Figure 8 also illustrates that the *EGR* almost stagnates for more than 15-20 location claims. This knowledge can be used to include a notion of freshness into the verification scheme. If an adversary is claiming the correct path in the beginning but lies about its track later on, the verification might work better if only the most recent 15 location claims are considered.

We can conclude that verification only works for tracks on which provers cover distances greater than the system's measurement error. For such tracks, we can say that the greater the distances covered by the prover, the less messages we need to verify tracks without false acceptances or rejections. To provide a real-world example for distances covered by potential provers, we again looked at data from the OpenSky network. Airplanes in the en-route airspace (i.e. at an altitude of about 30,000 ft) travel at a velocity of up to 300 m/s. That means that they cover distances of the order of kilometers within a few seconds, making them suitable provers for our track verification scheme.

2) **Global Verification Scheme:** In case the system has many verifiers covering an area of interest, our global scheme can be used to reduce false acceptances and rejections for small m. To gain insights on the global verification result, we conducted simulations similar to the previous ones. We placed a varying number of verifiers at random positions in a circular area with radius $r = 200$ m and used the same error model parameters (σ and σ_{drift}) as above. We picked this radius because it produces false rejections and acceptances with the local verification for $m \leq 13$ (see Figure 8). Compared to larger radii, this is a rather high number of least required messages for doing local verification without false rejections and/or acceptances. Thus, there is room for improvement which makes this radius illustrative for the benefits of the global scheme. Besides that and as mentioned above, distances on the order of hundreds of meters is realistic for location claims in aviation.

As before, we run our simulations for 1000 random tracks to derive max$_{honest}$ and min$_{dishonest}$ for each track length $|T|$ and number of verifiers $|V|$. Figure 9 shows the results. As in the local verification, tracks with just three location claims are still not properly verifiable due to the small number of samples for eliminating the noise. However, by increasing the number of verifiers to 7, we can perfectly verify tracks already after the fifth location claim with our global verification scheme.

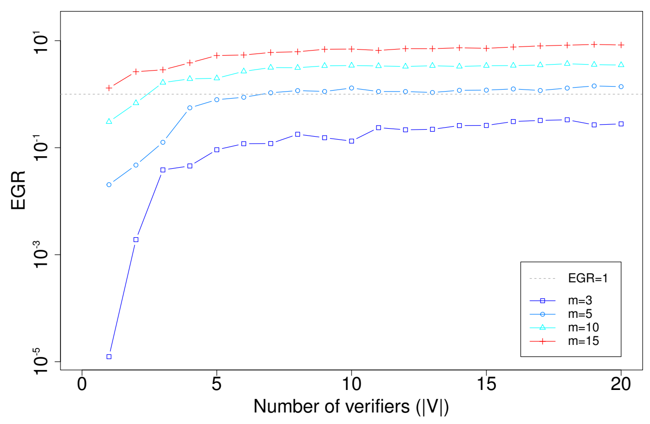

Fig. 9. Result of the simulation of the global verification scheme for 1000 tracks with an increasing number of verifiers and different numbers of received location claims ($m = |T|$). We used the following simulation parameters: $r = 200$ m, $\sigma = 50$ ns, $\sigma_{drift} = 20$ µs/s.

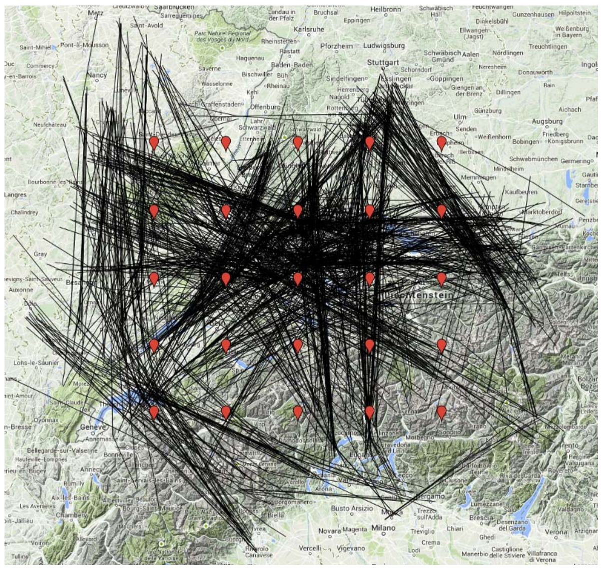

Fig. 10. The 1000 trajectories fetched from the OpenSky database for our applicability analysis. To achieve a good distribution of the verifiers across the considered area, we arranged 25 verifiers on a grid as marked by the bubbles.

Another result of this analysis is, that adding more than 7 verifiers does not result in a significant increase of the *EGR*.

VII. Applicability To Air Traffic

Besides analyzing error propagation, we conducted additional simulations with real flight tracks recorded by the OpenSky Network [14]. This enables us to assess the performance of our scheme with realistic tracks, proves the applicability to air traffic surveillance systems, and highlights challenges for a secure deployment of our scheme.

A. Simulation Setup & Data Preparation

The OpenSky Network is an Automatic Dependent Surveillance–Broadcast (ADS-B) sensor network which collects real-world air transportation communication data. In ADS-B, airplanes (and other vehicles) broadcast their position, velocity, and other status information periodically. The position reports, for instance, are broadcast twice per second. They contain the airplane's longitude, latitude, and altitude. Interpreting these position reports as location claims, ADS-B perfectly fits to our track verification scheme. The tracks are the trajectories of airplanes described by their ADS-B position reports.

For our analysis, we fetched 1000 flights from OpenSky's database which were received by one receiver. To have a large variety of trajectories, we selected a receiver close to Zurich Airport. This way, our simulations contain trajectories from the en-route airspace as well as from the approach area of the airport. We placed 25 verifiers in the reception area of the OpenSky receiver. To achieve a good distribution of verifiers across the reception area, we arranged the verifiers in a grid. Figure 10 shows the 1000 trajectories and the positions of the 25 verifiers.

While OpenSky provides timestamps with nanosecond precision for the time of arrival of the position reports, ADS-B does unfortunately not support timestamps for the transmission times. An attempt to estimate $\Delta_{i,j}$ for two position reports based on the airplane's reported velocity and the distance between the two reported positions failed due to the low resolution of velocity reports.

It is worth noting here that ADS-B has a feature in which transponders transmit position reports at discrete, known time intervals (see A.1.4.2.3.1 of [18]). This allows a receiver to estimate $\Delta_{i,j}$ very accurately without the need for explicit transmission timestamps. *Thus, our scheme is fully realizable within the ADS-B standard.* As of this writing, however, the ADS-B deployment is still in an initial phase. Too few airplanes support this mode at the moment and there are no guarantees and information on the accuracy of the current implementation as it is not yet certified for operational use.

Thus, we had to generate the timestamps $\bar{t}_i$ artificially to be able to apply our track verification scheme to the data. Therefore, we assumed that the track claimed by the airplane was correct and used the verifiers positions to estimate the propagation delay for each position report and verifier. Noise was added to these estimations in the same way as in the previous simulations from Section VI. This way, we were able to apply our scheme to trajectories with realistic properties such as dilution of precision and real shapes.

As in the previous simulations, the timestamps for the position reports were generated with random clock drifts with standard deviation $\sigma_{drift} = 20$ µs/s for each airplane and measurement error with $\sigma = 50$ ns for each timestamp. Then, we calculated the local verification results of all flights for each verifier. In order to gain insights on the time needed to verify a flight, we replayed 50 random position reports of each flight and recalculated $\mathcal{V}_T^x$ after the reception of each position report.

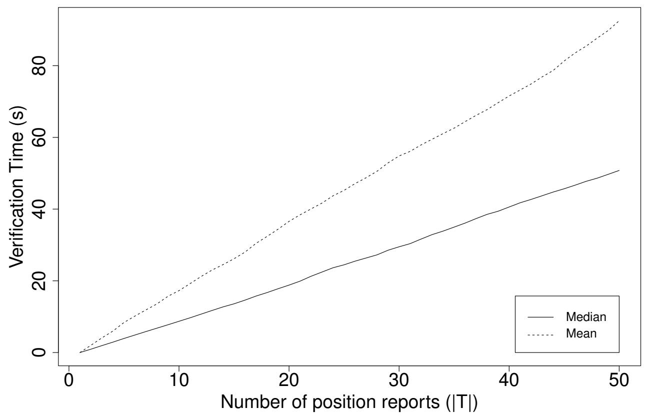

Fig. 11. Median and mean time needed to receive $|T|$ position reports of 1000 flights from OpenSky's database. The mean (median) arrival rate is 0.54 (0.95) messages per second.

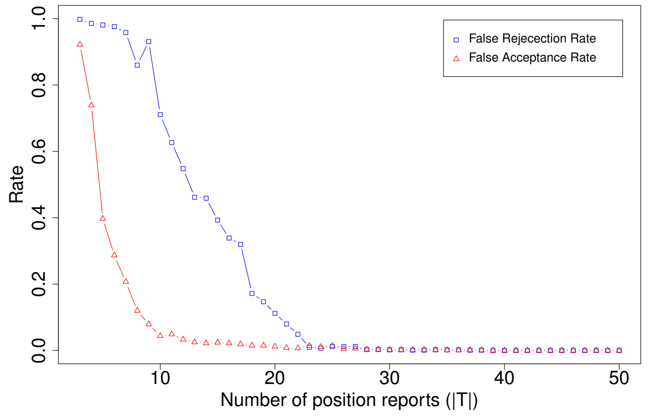

Fig. 12. False rejection rate for $\mathcal{T}_{local} = \min_{dishonest}$ and false acceptance rate for $\mathcal{T}_{local} = \max_{honest}$ of 1000 flights recorded by one receiver of the OpenSky Network.

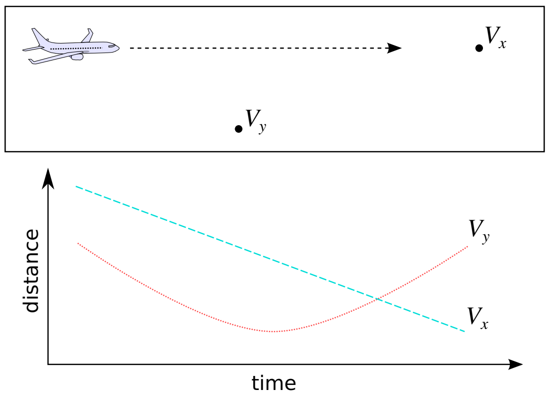

Fig. 13. Illustrative example scenario for a track which is highly linear for V_x and non-linear for V_y. The RMSE for V_x is very small because of the linear dependence of the distance on time. V_y cannot accurately approximate the distance with the linear model which results in a large RMSE.

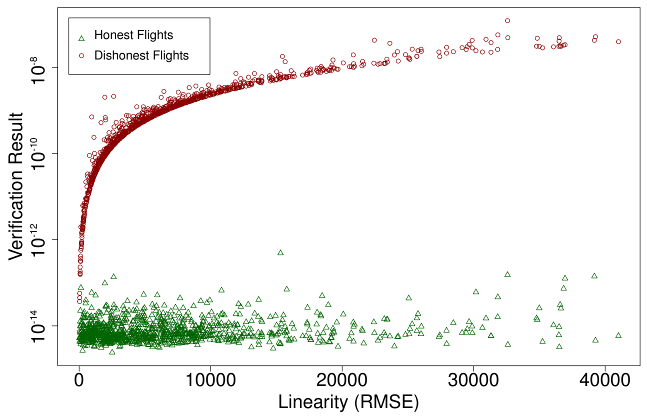

Fig. 14. The local verification result of 1000 real trajectories consisting of 50 position reports depending on their linearity. The less linear (i.e. the higher the RMSE), the greater the difference between honest and dishonest flights.

B. Results

1) **Verification Time:** The average and median verification time for different $|T|$ is shown in Figure 11. The ADS-B channel experiences high loss [2]. This loss results in a lower arrival rate than the transmission rate (which is two position reports per second). Altogether, the average message arrival rate was 0.54 and the median rate 0.95 messages per second. The difference in mean and median are a result of the high loss close to the edge of the receiver's reception range.

2) **Verification Result:** The false acceptance rate for $\mathcal{T}_{local} = \max_{honest}$ and the false rejection rate for $\mathcal{T}_{local} = \min_{dishonest}$ are shown in Figure 12. For example, if we set $\mathcal{T}_{local}$ such that all honest flights get accepted, 2.4% of the dishonest flights get falsely accepted after 15 messages. Conversely, setting $\mathcal{T}_{local}$ such that all dishonest flights get rejected, we observed a false rejection rate of 39.3%. Both, the false acceptance and false rejection rate dropped to zero after receiving 39 position reports.

A deeper analysis of the results revealed, that the false rejections and false acceptances are the result of some dishonest trajectories, which produce extremely small $\mathcal{V}_T^x$. The problem are particular trajectories, on which the change in distance to the receiving verifiers is monotonically and linearly increasing or decreasing. If this is the case, the deviation of the attacker's signal arrivals to the expected signal arrivals (i.e. $\Delta_i^x - \Delta_A^x$) is also monotonically and linearly decreasing or increasing since Δ_A^x is constant. The issue is, however, that clock drift also results in a linear deviation. As a consequence, our system cannot distinguish between these two deviations. Our drift estimator (Equation (8)) not only cancels out the clock drift, it also cancels out the linear deviation caused be the adversary's dishonesty. This results in very small $\mathcal{V}_T^x$ for dishonest tracks T with a high linearity. Flights, especially en-route flights, often have a shape close to a straight line with linearly changing Δ_i^x. This leads to the false acceptances in our simulations. An illustrative example for the linearity property of a trajectory is provided in Figure 13. The flight has a high linearity for V_x and a low linearity for V_y.

In order to further investigate this effect, we need an appropriate measure for the linearity property of tracks. Therefore,

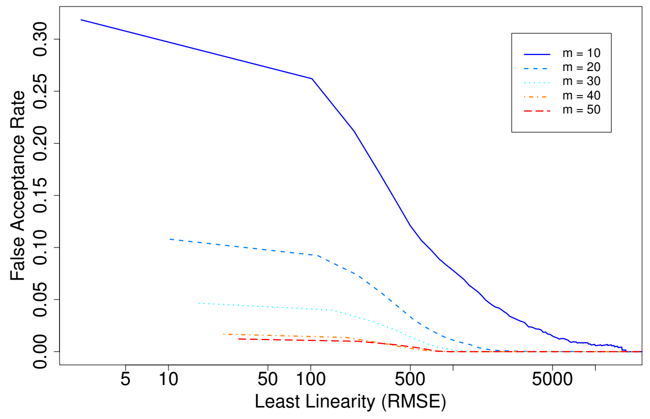

Fig. 15. The false acceptance rates for $\mathcal{T}_{local} = \max_{honest}$ and different $m = |T|$.

we quantified the linearity by doing a linear regression of the distances $\|\vec{p}_i - \vec{p}_x\|$ on times t_i. We then used the root-mean-square error (RMSE) as a measure for non-linearity: the higher the RMSE, the less linear the trajectory from the viewpoint of V_x. Figure 14 shows the dependency of the verification results of 1000 flights on the linearity. While the verification result stays constantly small for honest trajectory claims, the results for dishonest trajectories increases along with the non-linearity (i.e. with the RMSE).

We also considered the least required RMSE to achieve a zero false acceptance and rejection rate for the local verification scheme. The results are shown in Figure 15. We can conclude that the least required linearity becomes smaller the more position reports we used to calculate $\mathcal{V}_T^x$.

As this section showed, it is challenging to deal with linear tracks in combination with clock drift. This might pose a problem for some scenarios. For instance, the least number of required location claims increases if linear tracks are considered. This is disadvantageous for scenarios where fast detection rates are desired. To mitigate this problem, we discuss several approaches to avoid or deal with linearity in the next section.

C. Linearity Mitigation Approaches

Long reception ranges result in a high dilution of precision (DOP) for tracks far away from the verifiers. Due to DOP, distances between locations further away appear shorter which leaves less room for non-linearity. To tackle this problem, we recommend a *linearity-aware placement* of verifiers in the area of interest. For instance, distributing the available verifiers evenly across the area reduces the distances between tracks and verifiers and thus the DOP. In cases where only certain tracks are possible (e.g. provers are moving on roads or rails), a linearity-aware placement of verifiers can even prevent linear tracks completely.

Another way to deal with linearity is to reduce the least required RMSE by bounding the estimated clock drift. If the upper bound of clock drifts is known and $\hat{t}_{drift}^x$ exceeds it, the track claim might be dishonest. This approach is only applicable, if the clock drift bound is lower than the linear deviation caused by the dishonesty of the adversary.

Lastly, a collaborative scheme for estimating $\hat{t}_{drift}^x$ can also prevent attacks which exploit linearity. Therefore, the clock drift coefficients relative to some (or all) of the other verifiers must be determined. This knowledge can then be used to agree on a global $\hat{t}_{drift}^x$, making it impossible to hide linear dishonesty in different clock drift estimations. The pairwise clock drift coefficients can be determined using trusted provers. After a track with a sufficiently high non-linearity has been accepted by the system, they exchange their clock drift estimators and, by that, learn the clock drift coefficients for the other verifiers.

For cases in which none of the above approaches is feasible, verifiers must calculate the RMSE as part of the verification process. Each verifier can then assess whether it is in the position to verify a track or not. Tracks too linear for verification should be rejected a priori in order to avoid laying the system open to attacks.

VIII. Discussion & Future Work

The strength of our track verification scheme lies within its simplicity. Any node which knows its own position and is able to capture the timestamps of the received claims can calculate $\mathcal{V}_T^x$. Besides that, the scheme works completely passive. Except the track claims, there is no additional communication between verifiers and provers necessary. Hence, verifiers are simple devices which can be integrated into existing systems easily. Furthermore, they can run in parallel to systems which need to be secured without touching them.

This simplicity of our scheme enables many applications. In some of them, different threat models might be interesting. Therefore, we discuss several adjustments of our threat model. A deeper analysis will be subject to future work.

A. Mobile Adversary

In section III-A, we have proven that our scheme is secure for stationary adversaries by assuming a fixed propagation delay Δ_A^x. If we remove this assumption, the scheme might not be secure anymore. However, the adversary must be able to move in a way such that the propagation delays from the attackers positions to all verifiers change exactly as they would change on the claimed track. In scenarios where the adversary cannot move freely (e.g. due to obstacles in a city), a mobile adversary might not be able to claim arbitrary tracks. Yet, this might be a valid threat in other scenarios. For instance, if the attacker has more degrees of freedom than legitimate nodes, it might be able to successfully claim dishonest tracks. An example would be a vehicular ad hoc network and an adversary that uses a helicopter or drone to claim dishonest tracks. From a practical point of view, such an attack would still be hard to realize since such an adversary will most likely violate the reception area sanity check described in section V.

In [19], Tippenhauer et al. investigate the requirements for successful GPS spoofing attacks. They derive placement constraints for a mobile attacker which tries to spoof the GPS-derived location of several receivers. One of their results is that the attacker's transmission locations must lie on a hyperboloid to spoof two receivers and on a hyperbola for three receivers. In GPS, receivers use the TDoA of satellite signals to calculate their locations. Since our scheme also relies on ToA measurements, a mobile attacker would face similar placement constraints in our scheme. In fact, preliminary research on placement constraints suggests that the additional degree of freedom of the mobile attacker can be compensated by requiring an additional verifier.

B. Adversary's Knowledge

Another parameter of the threat model is the adversary's knowledge. In our security analysis, we assumed that the attacker knows everything. In particular, it is aware of its position and the positions of all verifiers. This knowledge makes the verification with $|V| < 3$ insecure.

Čapkun et al. proposed a scheme for secure location verification whose security is based on *covert base stations* (CBS) [13]. By CBS, the authors mean verifiers whose locations are not known to the attacker at the time of execution of the secure location verification. A potential attacker would have to guess the CBSs positions right in order to time the transmissions of its claims without causing inconsistencies at the verifiers.

This idea is also applicable to our scheme. It would benefit from CBSs in two ways. First, bypassing the reception area sanity check is much harder if the adversary does not know where its signal is expected. It could guess the area based on signal propagation models, but the effort would be much higher and the chance to be detected much larger. Second, the number of verifiers required to securely verify tracks can be reduced. In theory, just a single verifier would be sufficient as long as it is covered. If the attacker is not able to estimate the propagation delay Δ_A^x of its signal to verifier V_x, it can only guess Δ_A and would be detected with a high probability (see Figure 3).

Similar to CBSs, mobile verifiers would increase the detection probability of dishonest tracks significantly. The adversary must keep track of all verifiers to launch timing attacks and bypass the reception area sanity check. Mobile verifiers are indeed a realistic scenario. In air traffic monitoring, honest airplanes could also act as verifiers. Airplanes equipped with ADS-B receivers and GPS meet all requirements for calculating $\mathcal{V}_T^x$ for surrounding airplanes. If dishonest tracks are detected, the pilot could warn the ground stations, e.g., using voice communication. Another advantage of using airplanes as verifiers is that at high altitudes, airplanes can have ranges of more than thousand kilometers[4]. Together with the high density of today's air spaces, a world-wide coverage could be easily achieved without the need of new infrastructure. In OpenSky, a single sensor receives position reports of up to 60 airplanes at the same time during peak traffic hours. Thus, high numbers of verifiers $|V|$ could be achieved by using trusted airplanes for verification.

[4]assuming that communication is possible if there is a line-of-sight connection

C. Limits Of Our Scheme

As all location verification schemes which rely on signal arrival measurements, our scheme is not secure if an adversary is able to transmit independent signals to all verifiers. The attacker could use directional antennas or launch a coordinated attack from different positions. In that way, it can time the signal arrivals at the verifiers exactly as if they were sent from the claimed positions. However, such attacks are very sophisticated since they require an extremely accurate timing. In addition to that, the attacker still has to know the exact positions and reception ranges of all verifiers.

IX. Related Work

This section gives a brief overview of work that is related to ours. A more general overview of secure localization and secure location verification is available in [20].

1) **Distance bounding protocols:** Distance bounding protocols are two-way ranging protocols that rely on cryptographic techniques to enable a verifier to establish an upper bound on the physical distance to a prover. These protocols are based on timing the delay between sending out challenge bits and receiving back the corresponding response bits. The idea was first proposed by Brands and Chaum in 1994 [21]. Sastry et al. then proposed using it for secure location verification in 2003 [6]. Later, more general concepts for using distance bounding for secure location verification were proposed among others by Singelee and Preneel [7] and Čapkun and Hubaux [9]. Distance bounding protocols, however, are active and require highly specialized hardware to keep the processing time constant and as short as possible [8]. In contrast, our approach is passive and does not require any specialized hardware or protocol.

2) **Multilateration:** Mutlilateration is a passive localization technique based on the time difference of arrival (TDoA) of signals at geographically distributed stations. Multilateration is often used in radar systems for localization of mobile targets [22] and has also been proposed for location verification [5], [12]. The major drawback of multilateration is that it requires very precise time synchronization on the order of nanoseconds at the verifiers. This requirement makes the technique quite expensive and affordable only for smaller deployments. Conversely, secure track verification as we propose in this work does not require time synchronization which dramatically reduces the costs of infrastructure.

The authors of [16] have used mobility-differentiated ToA as a special form of TDoA for location surveying in sensor networks. However, the algorithms proposed in their work assume non-adversarial settings whereas our approach is designed to be secure against location spoofing attacks.

3) **Angle-of-arrival:** The location of wireless transmitters can be estimated by using the angle-of-arrival of the incoming signals. Systems have been proposed that rely on directional antennas [10], [11] to prevent attacks and to localize emitters.

However, directional antennas with a high degree of directionality are costly and several antennas are required at each verifier in order to capture attacks from all possible directions. In addition, angle-of-arrival methods are highly susceptible to multi-path reflections of the signal [23] and directional antennas typically exhibit side-lobes which can be exploited by the attacker to fool the system in real-world scenarios.

4) **Kalman Filters:** Kalman filters are algorithms that use a series of measurements observed over time to filter out noisy input data and produce more accurate estimations of the output state. Kalman filters have been proposed as a way to verify the integrity of the tracks received from ADS-B data [24]. However as pointed out by [25], [26], Kalman filters assume that the errors in the location claims are mainly because of non-malicious factors such as mobility or channel interference. Kalman filters are not secure against malicious injection of spoofed position claims.

X. Conclucion

In this work, we presented a mechanism for securely verifying tracks on which mobile nodes such as cars or aircraft claim to move on. Our scheme exploits the prover's mobility to avoid the need for synchronization and additional communication. Besides that, it does not assume any restrictions on the attacker's knowledge. We have proven that three verifiers using our scheme are able to securely detect stationary attackers. Furthermore, we have conducted extensive simulations to analyze the performance of our scheme under realistic conditions. We analyzed many different aspects such as error propagation, system requirements, and practicality. In particular, we used real traffic data to demonstrate its applicability to air traffic monitoring. Based on insights from our simulations, we derived requirements for a secure implementation of our scheme and discussed different threat models.

References

[1] T. Leinmuller, E. Schoch, and F. Kargl, "Position verification approaches for vehicular ad hoc networks," *IEEE Wireless Communications Magazine*, vol. 13, no. 5, pp. 16–21, 2006.

[2] M. Strohmeier, M. Schäfer, V. Lenders, and I. Martinovic, "Realities and Challenges of NextGen Air Traffic Management: The Case of ADS-B," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 111–118, May 2014.

[3] Andrei Costin and Aurélien Francillon, "Ghost is in the Air(traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices," Black Hat USA, Jul. 2012, white paper.

[4] M. Schäfer, V. Lenders, and I. Martinovic, "Experimental Analysis of Attacks on Next Generation Air Traffic Communication," in *Applied Cryptography and Network Security (ACNS)*. Springer, Jun. 2013.

[5] K. Pourvoyeur and R. Heidger, "Secure ADS-B usage in ATC tracking," in *Tyrrhenian International Workshop on Digital Communications - Enhanced Surveillance of Aircraft and Vehicles (TIWDC/ESAV)*. IEEE, Sep. 2014, pp. 35–40.

[6] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *Workshop on Wireless Security (WiSe)*. ACM, 2003.

[7] D. Singelee and B. Preneel, "Location verification using secure distance bounding protocols," in *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference (MASS)*. IEEE, Nov. 2005.

[8] K. B. Rasmussen and S. Čapkun, "Realization of RF Distance Bounding," in *USENIX Security Symposium*, 2010.

[9] S. Čapkun and J.-P. Hubaux, "Secure positioning of wireless devices with application to sensor networks," in *International Conference on Computer Communications (INFOCOM)*. IEEE, March 2005.

[10] L. Hu and D. Evans, "Using Directional Antennas to Prevent Wormhole Attacks ," in *Network and Distributed System Security Symposium (NDSS)*, Feb. 2004.

[11] L. Lazos, R. Poovendran, and S. Čapkun, "ROPE: Robust Position Estimation in Wireless Sensor Networks," in *ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*, Apr. 2005.

[12] A. Smith, R. Cassell, T. Breen, R. Hulstrom, and C. Evers, "Methods to Provide System-Wide ADS-B Back-Up, Validation and Security ," in *IEEE/AIAA Digital Avionics Systems Conference*, Oct. 2006.

[13] S. Čapkun, K. Rasmussen, M. Čagalj, and M. Srivastava, "Secure location verification with hidden and mobile base stations," *IEEE Transactions on Mobile Computing*, vol. 7, no. 4, pp. 470–483, Apr. 2008.

[14] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing Up OpenSky: A Large-scale ADS-B Sensor Network for Research," in *ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE Press, April 2014.

[15] T. Schmid, "Time in wireless embedded systems," Ph.D. dissertation, University of California, Los Angeles, 2009.

[16] J. Luo, H. V. Shukla, and J.-P. Hubaux, "Non-Interactive Location Surveying for Sensor Networks with Mobility-Differentiated ToA," in *International Conference on Computer Communications (INFOCOM)*. IEEE, April 2006.

[17] T. Witte and A. Wilson, "Accuracy of WAAS-enabled GPS for the determination of position and speed over ground," *Journal of Biomechanics*, vol. 38, no. 8, pp. 1717 – 1722, 2005.

[18] RTCA Inc., "Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance – Broadcast (ADS-B) and Traffic Information Services – Broadcast (TIS-B)," DO-260B with Corrigendum 1, Dec. 2011.

[19] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Čapkun, "On the requirements for successful gps spoofing attacks," in *ACM Conference on Computer and Communications Security (CCS)*, Oct. 2011.

[20] Y. Zeng, J. Cao, J. Hong, S. Zhang, and L. Xie, "Secure localization and location verification in wireless sensor networks: a survey," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 685–701, Jun. 2013.

[21] S. Brands and D. Chaum, "Distance-bounding protocols," in *Advances in Cryptology*, ser. EUROCRYPT. Springer, 1994.

[22] J. J. Herrero, J. Portas, J. C. R. C. Corredera, J. B. Portas, and F. Rodriguez, "ASDE and multilateration mode-S data fusion for location and identification on airport surface," in *IEEE Radar Conference*, Apr. 1999.

[23] S. Lakshmanan, K. Sundaresan, S. Rangarajan, and R. Sivakumar, "The myth of spatial reuse with directional antennas in indoor wireless networks," in *International Conference on Passive and Active Measurement (PAM)*, Apr. 2010.

[24] J. Krozel, D. Andrisani, M. Ayoubi, T. Hoshizaki, and C. Schwalm, "Aircraft ADS-B Data Integrity Check," in *AIAA Aviation, Technology, Integration, and Operations Conference (ATIO)*, Sep. 2004.

[25] K. Sampigethaya and R. Poovendran, "Visualization & assessment of ADS-B security for green ATM," in *IEEE/AIAA Digital Avionics Systems Conference (DASC)*, Oct. 2010.

[26] M. Strohmeier, V. Lenders, and I. Martinovic, "On the Security of the Automatic Dependent Surveillance-Broadcast Protocol," *IEEE Communications Surveys & Tutorials*, no. 99, 2014.