Discovery and Verification of Neighbor Positions in Mobile Ad Hoc Networks

Marco Fiore, *Member*, *IEEE*, Claudio Ettore Casetti, *Member*, *IEEE*,
Carla-Fabiana Chiasserini, *Senior Member*, *IEEE*, and Panagiotis Papadimitratos, *Member*, *IEEE*

Abstract—A growing number of ad hoc networking protocols and location-aware services require that mobile nodes learn the position of their neighbors. However, such a process can be easily abused or disrupted by adversarial nodes. In absence of a priori trusted nodes, the discovery and verification of neighbor positions presents challenges that have been scarcely investigated in the literature. In this paper, we address this open issue by proposing a fully distributed cooperative solution that is robust against independent and colluding adversaries, and can be impaired only by an overwhelming presence of adversaries. Results show that our protocol can thwart more than 99 percent of the attacks under the best possible conditions for the adversaries, with minimal false positive rates.

Index Terms—Neighbor position verification, mobile ad hoc networks, vehicular networks

1 Introduction

Location awareness has become an asset in mobile systems, where a wide range of protocols and applications require knowledge of the position of the participating nodes. Geographic routing in spontaneous networks, data gathering in sensor networks, movement coordination among autonomous robotic nodes, location-specific services for handheld devices, and danger warning or traffic monitoring in vehicular networks are all examples of services that build on the availability of neighbor position information.

The correctness of node locations is therefore an all-important issue in mobile networks, and it becomes particularly challenging in the presence of adversaries aiming at harming the system. In these cases, we need solutions that let nodes 1) correctly establish their location in spite of attacks feeding false location information, and 2) verify the positions of their neighbors, so as to detect adversarial nodes announcing false locations.

In this paper, we focus on the latter aspect, hereinafter referred to as *neighbor position verification* (NPV for short). Specifically, we deal with a mobile ad hoc network, where a pervasive infrastructure is not present, and the location data must be obtained through node-to-node communication. Such a scenario is of particular interest since it leaves the door open for adversarial nodes to misuse or disrupt the location-based services. For example, by advertising forged positions, adversaries could bias geographic routing or data gathering processes, attracting network traffic and then eavesdropping or discarding it. Similarly, counterfeit positions could grant adversaries unauthorized access to location-dependent services, let vehicles forfeit road tolls, disrupt vehicular traffic or endanger passengers and drivers.

In this context, the challenge is to perform, in absence of trusted nodes, a fully distributed, lightweight NPV procedure that enables each node to acquire the locations advertised by its neighbors, and assess their truthfulness. We therefore propose an NPV protocol that has the following features:

- It is designed for spontaneous ad hoc environments, and, as such, it does not rely on the presence of a trusted infrastructure or of a priori trustworthy nodes;
- It leverages cooperation but allows a node to perform all verification procedures autonomously. This approach has no need for lengthy interactions, e.g., to reach a consensus among multiple nodes, making our scheme suitable for both low- and high-mobility environments;
- It is reactive, meaning that it can be executed by any node, at any point in time, without prior knowledge of the neighborhood;
- It is robust against independent and colluding adversaries;
- It is lightweight, as it generates low overhead traffic.

Additionally, our NPV scheme is compatible with state-of-the-art security architectures, including the ones that have been proposed for vehicular networks [1], [2], which represent a likely deployment environment for NPV.

The rest of the paper is organized as follows: In Section 2, we review previous works, highlighting the novelty of our solution. In Section 3, we describe the system model, while the communication protocol, the objectives of the verification procedure and our main results are outlined in Section 4. The details of the NPV protocol and of verification tests are then presented in Section 5, and the

- *M. Fiore is with the CITI Laboratory, INRIA, INSA Lyon, Bat. Chappe, 6 Avenue des Arts, 69621 Villeurbanne Cedex, France. E-mail: marco.fiore@insa-lyon.fr.*
- *C.E. Casetti and C.-F. Chiasserini are with the Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy. E-mail: {casetti, chiasserini}@polito.it.*
- *P. Papadimitratos is with the School of Electrical Engineering at KTH, KTH Campus, Osquldas v. 10, 100 44 Stockholm, Sweden. E-mail: papadim@kth.se.*

Manuscript received 17 Mar. 2011; revised 26 Sept. 2011; accepted 18 Nov. 2011; published online 8 Dec. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2011-03-0143. Digital Object Identifier no. 10.1109/TMC.2011.258.

resilience of our solution to different attacks is analyzed in Section 6. Finally, we provide a performance evaluation of the protocol in a vehicular scenario in Section 7, and draw conclusions in Section 8.

2 Related Work

Although the literature carries a multitude of ad hoc security protocols addressing a number of problems related to NPV, there are no lightweight, robust solutions to NPV that can operate autonomously in an open, ephemeral environment, without relying on trusted nodes. Below, we list relevant works and highlight the novelty of our contribution. For clarity of presentation, we first review solutions to some NPV-related problems, such as secure positioning and secure discovery, and then we discuss solutions specifically addressing NPV.

Securely determining own location. In mobile environments, self-localization is mainly achieved through Global Navigation Satellite Systems, e.g., GPS, whose security can be provided by cryptographic and noncryptographic defense mechanisms [3]. Alternatively, terrestrial special-purpose infrastructure could be used [4], [5], along with techniques to deal with nonhonest beacons [6]. We remark that this problem is orthogonal to the problem of NPV. In the rest of this paper, we will assume that devices employ one of the techniques above to securely determine their own position and time reference.

Secure neighbor discovery (SND) deals with the identification of nodes with which a communication link can be established or that are within a given distance [7]. SND is only a step toward the solution we are after: simply put, an adversarial node could be securely discovered as neighbor and be indeed a neighbor (within some SND range), but it could still cheat about its position within the same range. In other words, SND is a subset of the NPV problem, since it lets a node assess whether another node is an actual neighbor but it does not verify the location it claims to be at. SND is most often employed to counter wormhole attacks [8], [9], [10]; practical solutions to the SND problem have been proposed in [11], while properties of SND protocols with proven secure solutions can be found in [12], [13].

Neighbor position verification was studied in the context of ad hoc and sensor networks; however, existing NPV schemes often rely on fixed [14], [15] or mobile [16] trustworthy nodes, which are assumed to be always available for the verification of the positions announced by third parties. In ad hoc environments, however, the pervasive presence of either infrastructure or neighbor nodes that can be aprioristically trusted is quite unrealistic. Thus, we devise a protocol that is autonomous and does not require trustworthy neighbors.

In [17], an NPV protocol is proposed that first lets nodes calculate distances to all neighbors, and then commends that all triplets of nodes encircling a pair of other nodes act as verifiers of the pair's positions. This scheme does not rely on trustworthy nodes, but it is designed for static sensor networks, and requires lengthy multiround computations involving several nodes that seek consensus on a common neighbor verification. Furthermore, the resilience of the protocol in [17] to colluding attackers has not been demonstrated. The scheme in [18] suits static sensor networks too, and it requires several nodes to exchange information on the signal emitted by the node whose location has to be verified. Moreover, it aims at assessing not the position but whether the node is within a given region or not. Our NPV solution, instead, allows any node to validate the position of all of its neighbors through a fast, one-time message exchange, which makes it suitable to both static and mobile environments. Additionally, we show that our NPV scheme is robust against several different colluding attacks. Similar differences can be found between our work and [19].

In [20], the authors propose an NPV protocol that allows nodes to validate the position of their neighbors through local observations only. This is performed by checking whether subsequent positions announced by one neighbor draw a movement over time that is physically possible. The approach in [20] forces a node to collect several data on its neighbor movements before a decision can be taken, making the solution unfit to situations where the location information is to be obtained and verified in a short time span. Moreover, an adversary can fool the protocol by simply announcing false positions that follow a realistic mobility pattern. Conversely, by exploiting cooperation among nodes, our NPV protocol is 1) reactive, as it can be executed at any instant by any node, returning a result in a short time span, and 2) robust to fake, yet realistic, mobility patterns announced by adversarial nodes over time.

The scheme in [21] exploits Time-of-Flight (ToF) distance bounding and node cooperation to mitigate the problems of the previous solutions. However, the cooperation is limited to couples of neighbor nodes, which renders the protocol ineffective against colluding attackers.

To our knowledge, our protocol is the first to provide a fully distributed, lightweight solution to the NPV problem that does not require any infrastructure or a priori trusted neighbors and is robust to several different attacks, including coordinated attacks by colluding adversaries. Also, unlike previous works, our solution is suitable for both low and high mobile environments and it only assumes RF communication. Indeed, non-RF communication, e.g., infrared or ultrasound, is unfeasible in mobile networks, where non-line-of-sight conditions are frequent and device-to-device distances can be in the order of tens or hundreds of meters. An early version of this work, sketching the NPV protocol and some of the verification tests to detect independent adversaries, can be found in [22].

3 System and Adversary Model

We consider a mobile network and define as *communication neighbors* of a node all the other nodes that it can reach directly with its transmissions [7]. We assume that each node knows its own position with some maximum error ϵ_p, and that it shares a common time reference with the other nodes: both requirements can be met by equipping communication nodes with GPS receivers.[1] In addition,

1. Small-footprint GPS receivers are commercially available, which achieve low synchronization and localization errors [23].

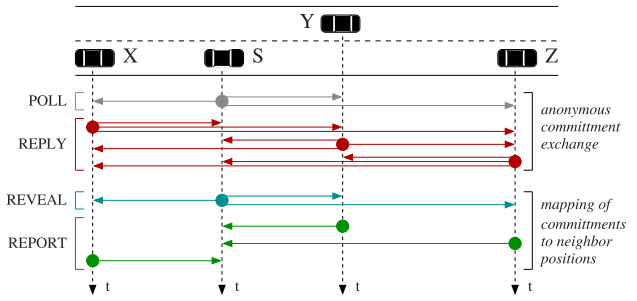


Fig. 1. Message exchange overview, during one instance of the NPV protocol.

nodes can perform Time-of-Flight-based RF ranging with a maximum error equal to ϵ_r. As discussed in [17], this is a reasonable assumption, although it requires modifications to off-the-shelf radio interfaces; also, promising techniques for precise ToF-based RF ranging have been developed [24].

We assume that node positions do not vary significantly during a protocol execution, since a complete message exchange takes no more than a few hundreds of milliseconds. The relative spatial movements of the nodes during such a period are taken into account through the tolerance value ϵ_m.

Nodes carry a unique identity[2] and can authenticate messages of other nodes through public key cryptography [27]. In particular, we assume that each node X owns a private key, k_X, and a public key, K_X, as well as a set of one-time use keys $\{k'_X, K'_X\}$, as proposed in emerging architectures for secure and privacy-enhancing communication [2], [25]. Node X can encrypt and decrypt data with its keys and the public keys of other nodes; also, it can produce digital signatures (Sig_X) with its private key. We assume that the binding between X and K_X can be validated by any node, as in state-of-the-art secure communication architectures [2], [26].

Nodes are *correct* if they comply with the NPV protocol, and *adversarial* if they deviate from it. As authentication essentially thwarts external adversaries, we focus on the more powerful internal ones, i.e., nodes that possess the cryptographic material to participate in the NPV and try to exploit it, by advertising arbitrarily erroneous own positions or inject misleading information. Internal adversaries cannot forge messages on behalf of other nodes whose keys they do not have. Thus, attacks against the cryptosystem are not considered, as correct implementation of cryptographic primitives makes them computationally infeasible.

We further classify adversaries into: *knowledgeable*, if at each time instant they know positions and (temporary) identities of all their communication neighbors, and *unknowledgeable*, otherwise; *independent*, if they act individually, and *colluding*, if they coordinate their actions.

4 Cooperative NPV: An Overview

We propose a fully distributed cooperative scheme for NPV, which enables a node, hereinafter called the *verifier*, to discover and verify the position of its communication neighbors. For clarity, here we summarize the principles of the protocol as well as the gist of its resilience analysis. Detailed discussions of message format, verification tests, and protocol resilience are provided in Sections 5 and 6.

2. This can be a permanent identifier or a temporary pseudonym, so as to ensure user privacy [25].

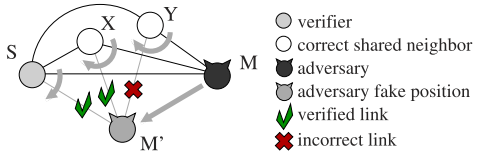


Fig. 2. Example of topological information stored by verifier S at the end of the message exchange and effect of a fake position announcement by M.

A verifier, S, can initiate the protocol at any time instant, by triggering the 4-step message exchange depicted in Fig. 1, within its 1-hop neighborhood. The aim of the message exchange is to let S collect information it can use to compute distances between any pair of its communication neighbors. To that end, POLL and REPLY messages are first broadcasted by S and its neighbors, respectively. These messages are anonymous and take advantage of the broadcast nature of the wireless medium, allowing nodes to record reciprocal timing information without disclosing their identities. Then, after a REVEAL broadcast by the verifier, nodes disclose to S, through secure and authenticated REPORT messages, their identities as well as the anonymous timing information they collected. The verifier S uses such data to match timings and identities; then, it uses the timings to perform ToF-based ranging and compute distances between all pairs of communicating nodes in its neighborhood.

Once S has derived such distances, it runs several position verification tests in order to classify each candidate neighbor as either:

1. *Verified*, i.e., a node the verifier deems to be at the claimed position;
2. *Faulty*, i.e., a node the verifier deems to have announced an incorrect position;
3. *Unverifiable*, i.e., a node the verifier cannot prove to be either correct or faulty, due to insufficient information.

Clearly, the verification tests aim at avoiding false negatives (i.e., adversaries announcing fake positions that are deemed verified) and false positives (i.e., correct nodes whose positions are deemed faulty), as well as at minimizing the number of unverifiable nodes. We remark that our NPV scheme does not target the creation of a consistent "map" of neighborhood relations throughout an ephemeral network: rather, it allows the verifier to independently classify its neighbors.

The basic principle the verification tests build upon is best explained by means of the example in Fig. 2. There, M is a malicious node announcing a false location M', so as to fraudulently gain some advantage over other nodes. The figure portrays the actual network topology with black edges, while the modified topology, induced by the fake position announced by M, is shown with gray edges. It is evident that the displacement of M to M' causes its edges with the other nodes to rotate, which, in turn, forces edge lengths to change as well. The tests thus look for discrepancies in the node distance information to identify incorrect node positions.

TABLE 1
Summary of Notations

Notation	*Description*
k_X (K_X)	private (public) key of X
k'_X (K'_X)	private (public) one-time key of X
t_X (t'_X)	actual (fake) transmission time of a message by X
t_{XY} (t'_{XY})	actual (fake) reception time at Y of a message by X
p_X (p'_X)	actual (fake) position of X
d_{XY}	distance between X and Y
ϵ_p (ϵ_r)	position (ranging) error
ϵ_m	tolerance to node movements during protocol execution
R	node proximity range
$\mathbb{N}_X$	current set of X's comm. neighbors
T_X	random wait interval after POLL reception at X
ρ_X	nonce sent by X
Sig_X	digital signature of X
C_X	certificate of X
$\mathfrak{c}_X$	*commitment* of X
i_X	temporary identifier assigned by S to X
$\mathbb{V}_X$	set of *verified* comm. neighbors of X
$\mathbb{U}_X$	set of *unverifiable* comm. neighbors of X
$\mathbb{F}_X$	set of *faulty* comm. neighbors of X
$\mathbb{W}_X$	set of *conditionally verified* comm. neighbors of X

A malicious node, knowing the protocol, can try to outsmart the tests in a number of different ways. Section 6 contains a comprehensive discussion of the protocol resilience, covering conceivable attack strategies that adversarial nodes could adopt. Overall, our analysis proves that:

- An unknowledgeable adversary has no possibility of success against our NPV protocol;
- An independent knowledgeable adversary M can *move* at most two links (with the verifier S and with a shared neighbor X) without being detected: however, any additional link (e.g., with another shared neighbor Y) leads to inconsistencies between distances and positions that allow to identify the attacker: this is the situation depicted in Fig. 2. In a nutshell, independent adversaries, although knowledgeable, cannot harm the system;
- Colluding knowledgeable adversaries can announce timing information that reciprocally validate their distances, and pose a more dangerous threat to the system. However, we prove that an overwhelming presence of colluders in the verifier neighborhood is required for an attack to be successful. Additionally, simulations in realistic scenarios prove the robustness of the NPV protocol even against large groups of colluding knowledgeable adversaries.

5 NPV Protocol

We detail the message exchange between the verifier and its communication neighbors, followed by a description of the tests run by the verifier. Table 1 summarizes the notations used throughout the protocol description.

5.1 Protocol Message Exchange

The value p_X is the current position of X, and $\mathbb{N}_X$ is the current set of its communication neighbors. We denote by t_X the time at which a node X starts a broadcast transmission and by t_{XY} the time at which a node Y starts receiving it. Note that these time values refer to the *actual* instant at which the node starts transmitting/receiving the first bit of the message at the physical layer. To retrieve the exact transmission and reception time instants, avoiding the unpredictable latencies introduced by interrupts triggered at the drivers level, a solution such as that implemented in [28] is required.[3] Furthermore, the GPS receiver should be integrated in the 802.11 card; software defined radio solutions combining GPS and 802.11 capabilities are proposed, among others, in [29], [30].

Now, consider a verifier S that initiates the NPV protocol. The message exchange procedure is outlined in Algorithm 1 for S, and in Algorithm 2 for any of S s communication neighbors.

Algorithm 1. Message exchange protocol: verifier.

1 **node** S **do**
2 $S \rightarrow * : \langle \text{POLL}, K'_S \rangle$
3 S : store t_S
4 **when** *receive* REPLY *from* $X \in \mathbb{N}_S$ **do**
5 S : store $t_{XS}, \mathfrak{c}_X$
6 **after** $T_{max} + \Delta + T_{jitter}$ **do**
7 $S : \mathbb{m}_S = \{(\mathfrak{c}_X, i_X) \mid \exists\, t_{XS}\}$
8 $S \rightarrow * : \langle \text{REVEAL}, \mathbb{m}_S, E_{k'_S}\{h_{K'_S}\}, Sig_S, C_S \rangle$

Algorithm 2. Message exchange protocol: any neighbor.

1 **forall** $X \in \mathbb{N}_S$ **do**
2 **when** *receive* POLL *by* S **do**
3 X : store t_{SX}
4 X : extract T_X uniform r.v. $\in [0, T_{max}]$
5 **after** T_X **do**
6 X : extract nonce ρ_X
7 $X : \mathfrak{c}_X = E_{K'_S}\{t_{SX}, \rho_X\}$
8 $X \rightarrow * : \langle \text{REPLY}, \mathfrak{c}_X, h_{K'_S} \rangle$
9 X : store t_X
10 **when** *receive* REPLY *from* $Y \in \mathbb{N}_S \cap \mathbb{N}_X$ **do**
11 X : store $t_{YX}, \mathfrak{c}_Y$
12 **when** *receive* REVEAL *from* S **do**
13 $X : \mathbb{t}_X = \{(t_{YX}, i_Y) \mid \exists\, t_{YX}\}$
14 $X \rightarrow S$:
$\langle \text{REPORT}, E_{K_S}\{p_X, t_X, \mathbb{t}_X, \rho_X, Sig_X, C_X\} \rangle$

POLL message. The verifier starts the protocol by broadcasting a POLL whose transmission time t_S it stores locally (Algorithm 1, lines 2-3). The POLL is anonymous, since 1) it does not carry the identity of the verifier, 2) it is transmitted employing a fresh, software-generated MAC address, and 3) it contains a public key K'_S taken from S's pool of anonymous one-time use keys that do not allow neighbors to map the key onto a specific node. We stress that keeping the identity of the verifier hidden is important in order to make our NPV robust to attacks (see the protocol analysis in Section 6). Since a source address has to be included in the MAC-layer header of the message, a fresh, software-generated MAC address is needed; note that this is considered a part of emerging cooperative systems [2], [25]. Including a one-time key in the POLL also ensures that the message is fresh (i.e., the key acts as a nonce).

3. This leads to a timing precision of around 23 ns, dictated by the 44 MHz clock of standard 802.11a/b/g cards. As mentioned above, we account for these errors through the ϵ_r parameter.

REPLY message. A communication neighbor $X \in \mathbb{N}_S$ that receives the POLL stores its reception time t_{SX}, and extracts a random wait interval $T_X \in [0, T_{max}]$ (Algorithm 2, lines 2-4). After T_X has elapsed, X broadcasts an anonymous REPLY message using a fresh MAC address, and locally records its transmission time t_X (Algorithm 2, lines 5-9). For implementation feasibility, the physical layer transmission time cannot be stamped on the REPLY, but it is stored by X for later use. The REPLY contains some information encrypted with S s public key (K'_S), specifically the POLL reception time and a nonce ρ_X used to tie the REPLY to the next message sent by X: we refer to these data as X's *commitment*, $\mathbb{C}_X$ (Algorithm 2, line 7). The hash $h_{K'_S}$, derived from the public key of the verifier, K'_S, is also included to bind POLL and REPLY belonging to the same message exchange.

Upon reception of a REPLY from a neighbor X, the verifier S stores the reception time t_{XS} and the commitment $\mathbb{C}_X$ (Algorithm 1, lines 4-5). When a different neighbor of S, e.g., Y, $Y \in \mathbb{N}_S \cap \mathbb{N}_X$, broadcasts a REPLY too, X stores the reception time t_{YX} and the commitment $\mathbb{C}_Y$ (Algorithm 2, lines 10-11). Since REPLY messages are anonymous, a node records all commitments it receives without knowing their originators.

REVEAL message. After a time $T_{max} + \Delta + T_{jitter}$, the verifier broadcasts a REVEAL message using its real MAC address (Algorithm 1, line 6). Δ accounts for the propagation and contention lag of REPLY messages scheduled at time T_{max}, and T_{jitter} is a random time added to thwart jamming efforts on this message. The REVEAL contains: 1) a map $\mathbb{m}_S$, that associates each commitment $\mathbb{C}_X$ received by the verifier to a temporary identifier i_X (Algorithm 1, line 7); 2) a proof that S is the author of the original POLL through the encrypted hash $E_{k'_S}\{h_{K'_S}\}$; 3) the verifier identity, i.e., its certified public key and signature (Algorithm 1, line 8). Note that using certified keys curtails continuous attempts at running the protocol by an adversary who aims at learning neighbor positions (i.e., at becoming knowledgeable) or at launching a clogging attack (see Section 6.4).

REPORT message. Once the REPORT message is broadcast and the identity of the verifier is known, each neighbor X that previously received S's POLL unicasts to S an encrypted, signed REPORT message. The REPORT carries X's position, the transmission time of X's REPLY, and the list of pairs of reception times and temporary identifiers referring to the REPLY broadcasts X received (Fig. 2, lines 12-14). The identifiers are obtained from the map $\mathbb{m}_S$ included in the REVEAL message. Also, X discloses its own identity by including in the message its digital signature and certified public key; through the nonce ρ_X, it correlates the REPORT to its previously issued REPLY. We remark that all sensitive data are encrypted using S s public key, K_S, so that eavesdropping on the wireless channel is not possible. At the end of the message exchange, *only the verifier knows all positions and timing information.* If needed, certified keys in REPORT messages allow the matching of such data and node identities (temporary or long-term, with the help of an authority if needed [2]).

5.2 Position Verification

Once the message exchange is concluded, S can decrypt the received data and acquire the position of all neighbors that participated in the protocol, i.e., $\{p_X, \forall X \in \mathbb{N}_S\}$. The verifier S also knows the transmission time t_S of its POLL and learns that of all subsequent REPLY messages, i.e., $\{t_X, \forall X \in \mathbb{N}_S\}$, as well as the corresponding reception times recorded by the recipients of such broadcasts, i.e., $\{t_{XY}, \forall X, Y \in \mathbb{N}_S \cup \{S\}\}$. Applying a ToF-based technique, S thus computes its distance from each communication neighbor, as well as the distances between all neighbor pairs sharing a link. More precisely, by denoting with c the speed of light, the verifier computes, for any communicating pair (X, Y) with $X, Y \in \mathbb{N}_S \cup \{S\}$, two distances: $d_{XY} = (t_{XY} - t_X) \cdot c$, from the timing information related to the broadcast message sent by X, and $d_{YX} = (t_{YX} - t_Y) \cdot c$, from the information related to the broadcast message by Y.

Once such distances have been computed, S can run the following three verification tests to fill the sets $\mathbb{F}_S$, $|\mathrm{V}_S$, and $\mathbb{U}_S$ with, respectively, faulty, verified and unverifiable nodes.

5.2.1 The Direct Symmetry Test (DST)

DST is the first verification performed by S and is detailed in Algorithm 3. There, $|\cdot|$ denotes the absolute value operator and $\|p_X - p_Y\|$ the euclidean distance between locations p_X and p_Y. In the **DST**, S verifies the direct links with its communication neighbors. To this end, it checks whether reciprocal ToF-derived distances are consistent 1) with each other, 2) with the position advertised by the neighbor, and 3) with a proximity range R. The latter corresponds to the maximum nominal transmission range, and upper bounds the distance at which two nodes can communicate. More specifically, the first check verifies that the distances d_{SX} and d_{XS}, obtained from ranging, do not differ by more than twice the ranging error plus a tolerance value ϵ_m (Algorithm 3, line 4), accounting for node spatial movements during the protocol execution. The second check verifies that the position advertised by the neighbor is consistent with such distances, within an error margin of $2\epsilon_p + \epsilon_r$ (Algorithm 3, line 5). Although trivial, this check is fundamental since it correlates positions to computed distances: without it, an attacker could fool the verifier by simply advertising an arbitrary position along with correct broadcast transmission and reception timings. Finally, as a sanity check, S verifies that d_{SX} is not larger than R (Algorithm 3, line 6). The verifier tags a neighbor as faulty if a mismatch is found in any of these checks,[4] since this implies an inconsistency between the position p_X and the timings announced by the neighbor (t_{SX}, t_X) or recorded by the verifier (t_{XS}, t_S).

Algorithm 3. Direct Symmetry Test (DST)

```
1 node S do
2   S : F_S ← ∅
3   forall X ∈ N_S do
4     if |d_SX − d_XS| > 2ε_r + ε_m or
5       |‖p_S − p_X‖ − d_SX| > 2ε_p + ε_r or
6       d_SX > R then
7       S : F_S ← X
```

4. The latter two checks are performed on both d_{SX} and d_{XS}, however in Algorithm 3 they are done on d_{SX} only, for clarity of presentation.

5.2.2 The Cross-Symmetry Test (CST)

In Algorithm 4, implements *cross* verifications, i.e., it checks on the information mutually gathered by each pair of communication neighbors. The **CST** ignores nodes already declared as faulty by the **DST** (Algorithm 4, line 5) and only considers nodes that proved to be communication neighbors between each other, i.e., for which ToF-derived mutual distances are available (Algorithm 4, line 6). However, pairs of neighbors declaring collinear positions with respect to S are not taken into account (Algorithm 4, line 7, where $line(p_X, p_Y)$ is the line passing by points p_X and p_Y). As shown in the next section, this choice makes our NPV robust to attacks in particular situations. For all other pairs (X, Y), the **CST** verifies the symmetry of the reciprocal distances (Algorithm 4, line 9), their consistency with the positions declared by the nodes (Algorithm 4, line 10), and with the proximity range (Algorithm 4, line 11). For each neighbor X, S maintains a link counter l_X and a mismatch counter m_X. The former is incremented at every new cross-check on X, and records the number of links between X and other neighbors of S (Algorithm 4, line 8). The latter is incremented every time at least one of the cross-checks on distance and position fails (Algorithm 4, line 12), and identifies the potential for X being faulty.

Algorithm 4. Cross-Symmetry Test (CST)

1 **node** S **do**
2 $S : \mathbb{U}_S \leftarrow \emptyset, \mathbb{W}_S \leftarrow \emptyset$
3 **forall** $X \in \mathbb{N}_S,\ X \notin \mathbb{F}_S$ **do**
4 $S : l_X = 0,\ m_X = 0$
5 **forall** $(X, Y) \mid X, Y \in \mathbb{N}_S,\ X, Y \notin \mathbb{F}_S,\ X \neq Y$ **do**
6 **if** $\exists\ d_{XY}, d_{YX}$ **and**
7 $p_S \notin line(p_X, p_Y)$ **then**
8 $S : l_X = l_X + 1,\ l_Y = l_Y + 1$
9 **if** $|d_{XY} - d_{YX}| > 2\epsilon_r + \epsilon_m$ **or**
10 $|\|p_X - p_Y\| - d_{XY}| > 2\epsilon_p + \epsilon_r$ **or**
11 $d_{XY} > R$ **then**
12 $S : m_X = m_X + 1,\ m_Y = m_Y + 1$
13 **forall** $X \in \mathbb{N}_S,\ X \notin \mathbb{F}_S$ **do**
14 **if** $l_X < 2$ **then** $S : \mathbb{U}_S \leftarrow X$
15 **else switch** $\frac{m_X}{l_X}$ **do**
16 **case** $\frac{m_X}{l_X} > \delta$ $S : \mathbb{F}_S \leftarrow X$
17 **case** $\frac{m_X}{l_X} = \delta$ $S : \mathbb{U}_S \leftarrow X$
18 **case** $\frac{m_X}{l_X} < \delta$ $S : \mathbb{W}_S \leftarrow X$

Once all neighbor pairs have been processed, a node X is added to the unverifiable set $\mathbb{U}_S$ if it shares less than two non-collinear neighbors with S (Algorithm 4, line 14). Indeed, in this case, the information available on the node is considered to be insufficient to tag the node as verified or faulty (see Section 6 for details). Otherwise, if S and X have two or more noncollinear common neighbors, X is declared as faulty, unverifiable, or *conditionally* verified, depending on the percentage of mismatches in the cross-checks it was involved in (Algorithm 4, lines 15-18). Specifically, X is added to $\mathbb{F}_S$ or $\mathbb{U}_S$, depending on whether the ratio of the number of mismatches to the number of checks is greater or equal to a threshold δ. If such a ratio is less than δ, X is added to a temporary set $\mathbb{WW}_S$ for conditionally verified nodes.

We point out that the lower the δ, the higher the probability of false positives, while the higher the δ, the higher the probability of false negatives. In the following, we set $\delta = 0.5$ so that the verifier makes a decision on the correctness of a node by relying on the opinion of the majority of shared (noncollinear) communication neighbors. As shown later, this choice makes our NPV highly resilient to attacks, unless the presence of adversaries becomes overwhelming.

5.2.3 The Multilateration Test (MLT)

MLT, in Algorithm 5, ignores nodes already tagged as faulty or unverifiable and looks for suspect neighbors in $\mathbb{WW}_S$. For each neighbor X that did not notify about a link reported by another node Y, with $X, Y \in \mathbb{WW}_S$, a curve $L_X(S, Y)$ is computed and added to the set $\mathbb{L}_X$ (Algorithm 5, lines 5-7). Such a curve is the locus of points that can generate a transmission whose Time Difference of Arrival (TDoA) at S and Y matches that measured by the two nodes, i.e., $|t_{XS} - t_{XY}|$. It is easy to verify that such a curve is a hyperbola, with foci in p_S and p_Y, and passing through the actual position of X.

Algorithm 5. Multilateration Test (MLT)

1 **node** S **do**
2 $S : \mathbb{V}_S \leftarrow \emptyset$
3 **forall** $X \in \mathbb{W}_S$ **do**
4 $S : \mathbb{L}_X \leftarrow \emptyset$
5 **forall** $(X, Y) \mid X, Y \in \mathbb{W}_S,\ X \neq Y$ **do**
6 **if** $\exists\ t_{XY}$ and $\nexists\ t_{YX}$ **then**
7 $S : \mathbb{L}_X \leftarrow L_X(S, Y)$
8 **forall** $X \in \mathbb{W}_S$ **do**
9 **if** $|\mathbb{L}_X| \geq 2$ **then**
10 $S :$
$p_X^{ML} = \arg\min_p \sum_{L_i, L_j \in \mathbb{L}_X} \|p - L_i \cap L_j\|^2$
11 **if** $\left\|p_X - p_X^{ML}\right\| > 2\epsilon_p$ **then**
12 $S : \mathbb{F}_S \leftarrow X,\ \mathbb{W}_S = \mathbb{W}_S \setminus X$
13 $S : \mathbb{V}_S = \mathbb{W}_S$

Once all couples of nodes in $\mathbb{WW}_S$ have been checked, each node X for which two or more unnotified links, hence two or more hyperbolas in $\mathbb{L}_X$, exist is considered as suspect (Algorithm 5, line 9). In such a case, S exploits the hyperbolae in $\mathbb{L}_X$ to multilaterate X s position, referred to as p_X^{ML}, similarly to what is done in [17] (Algorithm 5, line 10). Note that $\mathbb{L}_X$ must include at least two hyperbolae for S to be able to compute p_X^{ML}, and this implies the presence of at least two shared neighbors between S and X. p_X^{ML} is then compared with the position advertised by X, p_X (Algorithm 5, line 11). If the difference exceeds an error margin $2\epsilon_p$, X is moved to the faulty set $\mathbb{F}_S$. At the end of the test, all nodes still in $\mathbb{WW}_S$ are tagged as verified and moved to $|\mathrm{V}_S$ (Algorithm 5, lines 12-13).

6 Resilience Analysis

We analyze the robustness of our scheme against different types of internal adversaries. We classify the conceivable attacks into two classes, depending on the goal of the adversaries:

- Attacks where the adversaries aim at letting the verifier validate their own fake position;
- Attacks where the adversaries aim at disrupting the verification of correct node positions.

We focus on attacks of the first category in Sections 6.1 and 6.2, where we discuss the case of independent and colluding adversaries, respectively. In case of attacks of this first type, adversaries can tamper with the timing information in the REPLYs and REPORTs they generate, so that these confirm their false advertised locations. By considering the geometrical properties of the ToF-based ranging, we analyze the entire space of attacks against NPV. The effects of *combinations* of attacks of the first type is then investigated in our performance evaluation.

Attacks of the second category are analyzed in Section 6.3, where adversaries try to induce the verifier to tag a correct neighbor as faulty or unverifiable.

Finally, in Section 6.4 we discuss the robustness of our NPV scheme to generic attacks that are not specific to NPV. It is worth remarking that the NPV verification tests disregard nodes for which incomplete information is received, e.g., due to link or node failures. Such failures, when involving correct nodes, have the effect of degrading the number of nodes that corroborate other nodes legitimate claims. We have taken into account message losses in our simulation study of the protocol (Section 7). In this section, instead, we evaluate the NPV resilience level considering only the behavior of nodes participating in the whole message exchange, that is, for which the verifier has collected all the required information.

6.1 Faking Own Position: Independent Adversaries

We start by analyzing the attacks of the first type that can be launched by a single independent adversary in diverse network conditions, and explain the NPV protocol reactions they trigger. The discussion on the effects of the presence of multiple independent adversaries follows.

6.1.1 Basic Attack

In the simplest scenario, a verifier S runs the NPV protocol in presence of an adversary M, with which it shares no common neighbor. Let p'_M be the fake position that M advertises: as briefly mentioned above, M can announce a fake timing t'_{SM} in its REPLY, and a fake timing t'_M in its REPORT, so that p'_M is accepted by the verifier (i.e., $M \in |\mathrm{V}_S$).

More precisely, the **DST** run by S on M verifies that the reciprocal distances are consistent, i.e., that $|d_{SM} - d_{MS}| \leq 2\epsilon_r + \epsilon_m$, or:

$$\left|(t'_{SM} - t_S) \cdot c - (t_{MS} - t'_M) \cdot c\right| \leq 2\epsilon_r + \epsilon_m, \quad (1)$$

and that positions are also coherent with the distances, which implies $\left|\left\|p_S - p'_M\right\| - d_{SM}\right| \leq 2\epsilon_p + \epsilon_r$, or, equivalently:

$$\left|\left\|p_S - p'_M\right\| - (t'_{SM} - t_S) \cdot c\right| \leq 2\epsilon_p + \epsilon_r. \quad (2)$$

Therefore, the adversary must forge t'_M and t'_{SM}, so that (1)-(2) still hold after its real position p_M is replaced with p'_M. Solving the equation system obtained by setting the error margin to zero in (1)-(2) and expressing the ToF using the node positions, we obtain

$$t'_M = t_{MS} - \frac{\left\|p_S - p'_M\right\|}{c} = t_M + \frac{\|p_S - p_M\|}{c} - \frac{\left\|p_S - p'_M\right\|}{c} \quad (3)$$

$$t'_{SM} = t_S + \frac{\left\|p_S - p'_M\right\|}{c} = t_{SM} - \frac{\|p_S - p_M\|}{c} + \frac{\left\|p_S - p'_M\right\|}{c}. \quad (4)$$

Note that p'_M is chosen by M, and that M knows t_M in (3) (since this is the actual transmission time of its own REPLY) and t_{SM} in (4) (since this is the time at which it actually received S's POLL). Thus, we have a system of two equations in the two unknowns t'_M and t'_{SM}; M can solve it if it knows p_S. We call this forging of transmission and reception timings with respect to S the **basic** attack.

We stress that, in order to know p_S, M must be a knowledgeable adversary, which implies two conditions: first, M must have previously run the NPV protocol to discover the identity and position of its neighbors; second, the position of the verifier must have not changed since such discovery procedure. Clearly, as M cannot foresee when S starts the NPV protocol, such a condition is not easy to fulfill, especially in highly mobile environments. Nevertheless, if aware of S's location, M could successfully run a **basic** attack, provided that the advertised position p'_M is within the proximity range R. As a consequence, the NPV marks isolated neighbors as unverifiable in the **CST**.

Let us now add to the previous scenario $n \geq 1$ nodes, $X_1, \ldots, X_n$, which are correct neighbors common to S and M. The discussion above still holds, since the fake position advertised by M must still pass the **DST**. Thus, M has to know S s current position and to forge t'_M and t'_{SM} according to p_S and p'_M, as in (3)-(4). However, the presence of common neighbor(s) introduces two additional levels of security, which make the **basic** attack ineffectual.

First, the POLL and REPLY messages are anonymous; hence, upon their reception, even a knowledgeable M does not know which node among $S, X_1, \ldots, X_n$ is the verifier. Nevertheless, in order to take part in the protocol, M is forced to advertise the fake POLL reception time t'_{SM} in its REPLY, before receiving the REVEAL and discovering the identity of the verifier. The only option for M is then to randomly guess who the verifier is and properly change t_{SM} into t'_{SM}, as in (4). This implies a probability of success in guessing the actual sender of the POLL equal to $1/(n+1)$.

Second, in presence of shared (noncollinear) neighbors, S can run the **CST** on the (M, X_i) pairs, with $i = 1, \ldots, n$. As the **basic** attack only forges messages transmission and reception timings with respect to S, the fake position p'_M will present discrepancies with the reciprocal reception times of REPLY messages at M and X_i. This will result in a **CST** failure, revealing and thus preventing the attack.

6.1.2 REPLY-Disregard Attack

Whenever there are n neighbors $X_1, \ldots, X_n$ common to S and M, a possible strategy for the adversary, provided that it correctly guesses the identity of the verifier, is *not* to announce one or more of the common neighbors. That is, M will not include the (t_{X_iM}, i_{X_i}) data in its REPORT, thus deliberately denying to have received X_i's REPLY. We name this **REPLY-disregard** attack.

It follows that S cannot perform a cross-check on the pairs (M, X_i) in the **CST**. However, a REPLY-**disregard** attack

does not bring any significant advantage to M. Indeed, the exclusion of (some or all of) the common neighbors reduces the system to one of the scenarios discussed for the **basic** attack, hence the adversary is at most tagged as unverifiable by S. More importantly, the maximum number of REPLYs an adversary can disregard is exactly one, otherwise it is classified as faulty in the **MLT**.

6.1.3 Hyperbola-Based Attack

This attack again attempts to fool the **CST**. More specifically, it scales up the **basic** attack by also forging the timings relative to the shared neighbor(s).

First, consider that S and M share a noncollinear common neighbor X. The **CST** on the (M, X) pair at S requires that $|d_{XM} - d_{MX}| \leq 2\epsilon_r + \epsilon_m$ and $|\|p_X - p_M\| - d_{XM}| \leq 2\epsilon_p + \epsilon_r$. Applying the same substitutions as in (3) and (4), this means that M is forced to advertise the following fake timings:

$$t'_M = t_M + \frac{\|p_X - p_M\|}{c} - \frac{\|p_X - p'_M\|}{c}, \quad (5)$$

$$t'_{XM} = t_{XM} - \frac{\|p_X - p_M\|}{c} + \frac{\|p_X - p'_M\|}{c}. \quad (6)$$

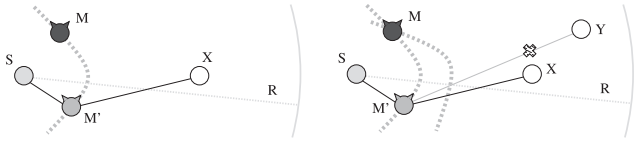
If M is knowledgeable, and thus aware of X's current position p_X, it can solve (6) and announce the forged t'_{XM} in its REPORT to S. However, (5) introduces a second expression for t'_M, while M can advertise only one t'_M in its REPORT. In order to pass both the **DST** and the **CST**, M needs to announce a t'_M that satisfies (3) and (5), which implies

$$\|p_S - p_M\| - \|p_S - p'_M\| = \|p_X - p_M\| - \|p_X - p'_M\|. \quad (7)$$

In other words, M is constrained to choose locations with the same distance increment (or decrement) from S and X. In (7), p_S, p_X, and p_M are fixed and known, hence the distances between p_S and p_M, and between p_X and p_M, can be considered as constant. We thus rewrite (7) as $\|p_X - p'_M\| - \|p_S - p'_M\| = k$. This is the equation, with the unknown p'_M, of a hyperbola with foci in p_S and p_X that passes through p_M. It follows that only positions p'_M on such hyperbola can satisfy the four constraints in (3), (4), (5), and (6). As a conclusion, the **hyperbola-based** attack consists in advertising a fake position that lies on the aforementioned curve, as well as message transmission and reception times that validate such a position. An example is provided in Fig. 3a.

Note that, in order to successfully perform a **hyperbola-based** attack, an adversary has to 1) know the position of both S and X, 2) correctly guess the identity of the verifier, and 3) advertise a fake position only along a specific curve. Although these are restrictive conditions, the **CST** still marks as unverifiable the nodes that passed the **DST** but share only one neighbor with the verifier, so as to avoid any possibility of successful **hyperbola-based** attack.

We now consider a second correct, noncollinear node $Y \in \mathbb{N}_S \cap \mathbb{N}_M$, and show that, in such a scenario, also the **hyperbola-based** attack becomes futile. Note that no assumption is made on the connectivity between the two neighbors X and Y. By extending the previous reasoning, in presence of two common correct neighbors, X and Y, M has to forge four time values, i.e., t'_M, t'_{SM}, t'_{XM}, and t'_{YM}, so



(a) One shared neighbor (b) Two shared neighbors

Fig. 3. Hyperbola-based attack: (a) If it correctly guesses S's identity, a knowledgeable adversary M can forge timings with respect to S and X (black lines), so that they agree with any fake position M' lying on the hyperbola with foci in S, X, and passing by M. M is unverifiable. (b) M can only forge timings in agreement with fake positions that lie on both hyperbolae of foci in S, X, and S, Y, and passing by M. The only such point, i.e., the intersection, matches M s actual position. When moving on the hyperbola of foci S and X, the timing with respect to Y (crossed gray line) is not verified.

that six equations are satisfied, i.e., (3), (4), (5), (6) and two additional equations[5] corresponding to the cross-check with the second common neighbor Y. This implies that the fake REPLY transmission time t'_M announced by M must now fulfill three constraints, or, equivalently, M must advertise a position p'_M that is equally farther from (or closer to) S, X, and Y with respect to its actual location p_M. The *only* point satisfying such a condition lies at the intersection of three hyperbolae with foci in p_S and p_X, p_S and p_Y, p_X, and p_Y, respectively, and it corresponds to the real position of the adversary, p_M. In other words, if it shares two neighbors with S, an adversary cannot successfully claim to be at any location other than its actual one, not even if it is knowledgeable, it correctly guesses the role of all other nodes, and it performs a **hyperbola-based** attack. An example is provided in Fig. 3b. We also stress that the presence of additional shared neighbors simply introduces other constraints on t'_M, and thus further binds M to its actual position.

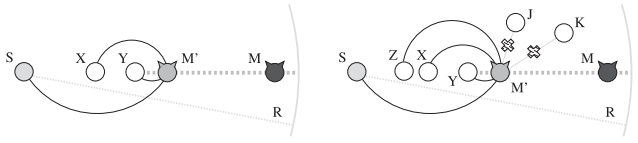
Similarly, combining a **hyperbola-based** attack with a **REPLY-disregard** attack yields no chance of success. As a matter of fact, ignoring REPLY messages from one or multiple shared neighbors results in reverting the system to one of the cases previously analyzed, with the adversary being tagged at best as unverifiable.

6.1.4 Collinear Attack

The above discussion shows that the presence of two or more correct common neighbors, which can be used to perform the cross-checks in the **CST**, is a condition that foils all the attack strategies introduced so far. There exists however a last type of attack, which we name **collinear** attack, that we need to discuss.

The **collinear** attack builds on the following geometrical property: if three points are collinear (i.e., lie on the same line), the hyperbola having as foci two of the points (one of which is that in the middle) and passing by the third degenerates into the half-line originating at the intermediate point and passing by the third one. This property implies that, if two shared neighbors X and Y lie between S and the adversary M, and all four nodes are collinear, the hyperbolae that pin the actual position of the adversary p_M degenerate to partially overlapping half-lines. This allows M to forge timings relative to S, X, and Y consistent

5. The latter two equations can be obtained from (5)-(6) by replacing p_X, t_{XM}, and t'_{XM}, respectively, with p_Y, t_{YM}, and t'_{YM}.



(a) Two shared neighbors (b) Multiple shared neighbors

Fig. 4. Collinear attack: (a) The hyperbola of foci in S, X (Y) and passing by M degenerates into a half-line with origin in X (Y). The hyperbolae intersection region, over which M can announce a fake position M' with correct timings, becomes the segment originating at Y and bounded by R (dashed gray line). M is unverifiable. (b) M can announce timings that are consistent with its fake position M' with respect to the three collinear neighbors X, Y, and Z (black lines). This is not possible with respect to J and K (crossed-out gray lines). As a countermeasure, collinear neighbors are not cross-checked in the **CST**. M is tagged as faulty.

with any fake position over the segment originating at the shared collinear neighbor that is closer to M, passing by p_M and bounded by R. An example is given in Fig. 4a.

In the more general case of any number of common neighbors to S and M, the **collinear** attack would allow an adversary to appear correct to the shared neighbors that are collinear with it and S, as in Fig. 4b. Again, this requires the adversary to be knowledgeable, to correctly guess the origin of POLL and REPLY messages it receives, and to limit the choice of its fake position to a specific segment.

As a countermeasure to **collinear** attacks, in the **CST** S discards pairs of neighbors that announce collinear positions with it (Algorithm 4, line 7). When collinear neighbors are dropped, a **collinear** attack results, for instance, in the adversary being tagged as unverifiable in Fig. 4a (since there are no noncollinear shared neighbors) and as faulty in Fig. 4b (since there are two noncollinear shared neighbors).

We remark that, on the one hand, not allowing cross-checks that involve collinear nodes prevents **collinear** attacks. On the other, it reduces the number of correct neighbors that can contribute to identifying adversarial nodes. However, as shown in Section 7, this approach ensures high resilience to attacks as well as reliability in identifying correct nodes as verified.

6.1.5 Multiple Independent Adversaries

Multiple independent adversaries in the neighborhood of the verifier just damage each other, by announcing false positions that reciprocally spoil the time computations discussed in the previous sections. Thus, all cross-checks on pairs of noncolluding adversaries result in mismatches in the **CST**, increasing their chances to be tagged as faulty by the verifier.

Where multiple independent adversaries can harm the system is in the verification of correct neighbors. As a matter of fact, a node is tagged as verified if it passes the strict majority of cross-checks it undergoes. A correct node surrounded by several adversaries can thus be marked as faulty (unverifiable), if it shares with the verifier a number of noncollinear independent adversaries greater than (equal to) the number of noncollinear correct nodes. However, situations where a correct node shares mostly uncoordinated adversarial neighbors with the verifier are unlikely to occur in realistic scenarios, as also shown by our performance evaluation.

6.1.6 Summary

We conclude that a single independent adversary cannot perform any successful attack against the NPV scheme. Indeed, in presence of a limited number of noncollinear neighbors in common with the verifier, a knowledgeable adversary can attempt one of the strategies outlined before, but it is tagged at most as unverifiable. When the shared neighborhood increases in size, the probability that the adversary is tagged as faulty rapidly grows to 1. Multiple independent adversaries can only harm each other, thus reducing their probability of successfully announcing a fake position.

6.2 Faking Own Position: Colluding Adversaries

We assume that colluders share out-of-band links with negligible latency, through which they exchange information, and can perform complex distributed computations. This notwithstanding, in the following we show that our scheme is resistant to coordinated attacks[6] as well, unless the presence of colluding adversaries in the neighborhood of the verifier becomes overwhelming.

6.2.1 Basic Attack

The simplest way adversarial nodes can cooperate to make the verifier S trust the fake positions they announce is by extending the **basic** attack introduced in Section 6.1.1. More precisely, other than individually announcing POLL reception timings that agree with their fake positions, colluding adversaries can mutually validate the false information they generate. They can forge the reception times of reciprocal REPLY messages, so that all cross-checks in the **CST** involving the colluders are passed. A perfect cooperation thus results in the colluding adversaries ability to alter all distances between them without being noticed.

We remark that the adversaries still need to know S's position in order to compute and advertise timings that confirm their fake position. This time, however, if at least three adversaries cooperate to perform the attack, they do not need to be knowledgeable. As a matter of fact, they can exploit their real positions and POLL reception times to multilaterate the coordinates of the verifier.

Our NPV correctly identifies such a **basic** attack through the **CST**, as long as the majority of the (noncollinear) neighbors shared by S and an adversary are not colluding with the latter. An example is shown in Fig. 5.

6.2.2 REPLY-Disregard Attack

As in the case of independent adversaries, multiple colluders can gain advantage by ignoring REPLY messages from noncolluding nodes. This lets them avoid cross-checks that could result in mismatches, and, eventually, in being tagged as faulty. Indeed, $n \geq 3$ colluders could disregard the REPLY received from all noncolluding nodes and still advertise a number $n-1 \geq 2$ of neighbors that would report consistent timings with theirs—a sufficient condition to pass the **CST**. This behavior, however, is properly handled in our NPV by the **MLT**, which tags as faulty a neighbor that disregarded (intentionally or not) two or more REPLY messages and announced a location other than the multilaterated one.

6. Note that, since our NPV exploits ToF-based ranging, wormhole attacks reduce to Sybil attacks, as discussed in Section 6.4.

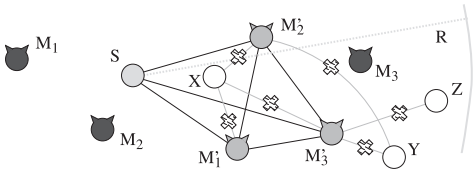


Fig. 5. Cooperative basic attack. The colluding adversaries M_1, M_2, and M_3 can forge timings that validate their fake positions M_1', M_2', and M_3' with respect to S, as well as to each other (black lines). However, cross-checks with noncolluding neighbors fail (crossed-out gray lines). M_1, sharing with S the other two colluders and X, is tagged as verified. M_2, sharing with S the other two colluders and X, Y, is unverifiable. M_3, sharing with S the other two colluders and X, Y, Z, is marked as faulty.

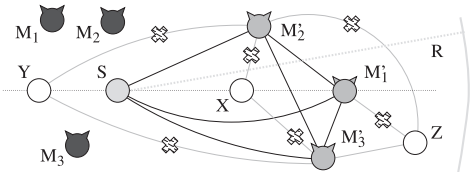


Fig. 7. Cooperative collinear attack. The adversary M_1 announces a fake position M_1' that is collinear with the verifier S and the noncolluding neighbors X, Y, avoiding cross-checks with the latter two. M_2 and M_3, colluding with M_1, declare noncollinear false locations (M_2', M_3'), thus guaranteeing to M_1 a majority of validated cross-checks. The adversary M_1 is tagged as verified, while M_2 and M_3 are marked as faulty.

6.2.3 Hyperbola-Based Attack

The colluding attackers agree not only on the position of the verifier (either guessed or multilaterated), but also pick a noncollinear common neighbor, X, that they share with S: each colluder then computes the hyperbola with foci S, X, and passing through its own real position, and announces a fake location on such a curve. This allows the adversaries to announce correct links 1) with the verifier, 2) among themselves, and 3) with the selected neighbor X, which becomes an involuntary ally in the attack. Again, the location of X must be randomly guessed by two colluders, while it can be multilaterated by three or more cooperating adversaries.

In presence of such a **hyperbola-based** attack, the **CST** correctly tags an adversary M as faulty if the noncollinear common neighbors between the verifier and M that do not collude with M outnumber the colluding ones by 3. Note that the two additional correct neighbors are required to counter the effect of X unintentionally taking part into the attack. An example is depicted in Fig. 6.

6.2.4 Collinear Attack

Unlike independent adversaries (Section 6.1.4), multiple colluders can take advantage of a **collinear** attack. In particular, one or more adversaries can *purposely* announce positions that are collinear with those of some noncolluding neighbors, so as to avoid cross-checks with them. Then, they can rely on colluders that declared noncollinear positions to pass the **CST**, as in Fig. 7.

In other words, colluders can launch a **collinear** attack as a *legal* mean to avoid unwanted neighbors. Such a gain comes at the cost of a restricted freedom of movement, since the fake position must lie on a specific segment (see Section 6.1.4). The robustness of our NPV to this kind of attacks depends on the network layout: environments where nodes tend to form straight topologies (such as vehicular ones) are more prone to suffer from **collinear** attacks. In general, the NPV is resistant to **collinear** attacks as long as the majority of the shared neighbors are not colluding or collinear.

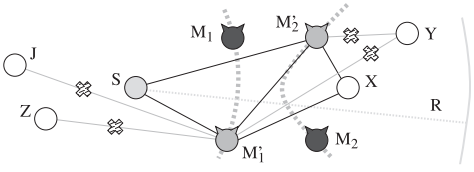


Fig. 6. Cooperative hyperbola-based attack: If two colluding adversaries, M_1 and M_2, are knowledgeable and correctly guess the identity of S, they can forge timings with respect to S and X as well as to each other (black lines) and announce the fake positions M_1' and M_2', respectively. M_1' (M_2') can be any point on the hyperbola with foci in S, X, and passing by M_1 (M_2). However, M_1 shares four correct and one colluding, noncollinear neighbors with S, while M_2 shares two correct and one colluding, noncollinear neighbors with S, hence M_1 is tagged as faulty and M_2 is tagged as verified.

6.2.5 Summary

As a conclusion on coordinated attacks, it is the nature of the neighborhood that determines the performance of the NPV scheme in presence of colluders. However, the simulation results in Section 7 show that, in realistic environments, our solution is very robust even to attacks launched by large groups of knowledgeable colluders.

6.3 Discrediting Other Neighbors

A different objective of the adversary can be to discredit other nodes by inducing the verifier to tag them as faulty or unverifiable. To this end, an adversary M needs to announce a fake timing t'_{XM} (i.e., the time at which M claims it has received X's REPLY), for any neighbor $X \in \mathbb{N}_S \cap \mathbb{N}_M$ that it wants to discredit. By doing so, M can disrupt the cross-checks made by S on the pair (X, M) in the **CST**. When launched by a single adversary, such an attack can succeed if there are only two additional correct nodes (which are tagged as unverifiable by S). In all other configurations, a single adversary cannot affect the assessment of other correct nodes.

When launched by multiple adversaries, no matter whether they are independent or colluding, the effect of this attack is the same as the one highlighted in Section 6.1.5. We recall that our NPV protocol provides protection to a correct node X, as long as the number of adversarial neighbors it shares with the verifier S is lower than that of correct common neighbors. Vice versa, if the number of adversarial shared neighbors trying to discredit X is greater than (equal to) the number of correct common neighbors, X is tagged as faulty (unverifiable).

6.4 Other Attacks

6.4.1 Jamming

This is the only external attack that can harm the system. Any adversary (internal or external) can jam the channel and erase REPLY or REPORT messages. However, to succeed, M should jam the medium continuously for a long time, since it cannot know when exactly a node will

transmit its REPLY or REPORT. Or, M could erase the REVEAL, but, again, jamming should cover the entire T_{jitter} time. Overall, there is no easy point to target: a jammer has to act throughout the NPV execution, which implies a high energy consumption and is a disruptive action possible against any wireless protocol. In addition, mobility makes it harder to repeatedly jam different instances of the NPV protocol run by the same verifier.

6.4.2 Clogging

An adversary could initiate the NPV protocol multiple times in a short period and get repeated REPLY and REPORT messages from other nodes, so as to congest the channel. In particular, REPORTs are larger in size, thus likely cause the most damage. However, NPV has a way of preventing that: the initiator must unveil its identity before such messages are transmitted by neighbors. An exceedingly frequent initiator can be identified, and its REVEALs ignored, thanks to the use of certified keys. REPLYs instead are small in size and are broadcast messages (thus require no ACK): their damage is limited, but their unnecessary transmission is much harder to thwart. Indeed, REPLY messages are sent after an anonymous POLL; such an anonymity is a hard-to dismiss requirement, since it is instrumental for keeping the identity of the verifier hidden. As a general rule, correct nodes can reasonably self-limit their responses if POLLs arrive at excessive rates.

6.4.3 Adversarial Use of Directional Antennas

Assume that adversaries are equipped with directional antennas and multiple radio interfaces. As a correct node S starts the NPV protocol, a knowledgeable adversary M can send different REPLYs through each interface at different time instants: any correct neighbor X would record a time t'_{MX}, compliant with the fake position p'_M, allowing M to pass the cross-check with X in the **CST**. If M can fool a sufficient number of neighbors, it is tagged as verified. However, M needs as many directional antennas and radio interfaces as the number of neighbors it wants to fool, hoping that no two such neighbors are within the beam of the same antenna. The complexity, cost, and chances of failure make this attack hardly viable. We also remark that, since our approach exploits ToF-based ranging, such an attack cannot be launched by using a steerable antenna, which takes an exceedingly long time to swap from one sector to another.

6.4.4 Sybil and Relay (Wormhole) Attacks

An adversary can assume several trusted identities, $\mathcal{M} = \{M_1, \ldots, M_I\}$, if 1) it owns several certificated pairs of public/private keys (Sybil attack), or 2) it impersonates colluding adversaries at the end of wormholes. The availability of several identities could be used by an adversary to acquire its neighbor positions, i.e., to become knowledgeable. However, as shown in Section 6.1, attacks launched by independent, knowledgeable adversaries have no chance of success. Furthermore, by announcing timings that are consistent among the identities in $\mathcal{M}$, the adversary can behave as a group of colluders of size I. The analysis in Section 6.2 thus applies to such attacks as well, except for the fact that the adversary cannot acquire the position of other nodes through triangulation. A verifier suspecting[7] that this attack is being launched can run the **MLT** to determine whether messages from nodes in $\mathcal{M}$ come from the same location.

7 Performance Evaluation

We evaluated the performance of our NPV protocol in a vehicular scenario. Results obtained in a pedestrian scenario are available as supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC. 2011.258.

We focus on knowledgeable adversaries whose goal is to make the verifier believe their fake positions, and we describe the *best attack strategy* they can adopt in Section 7.1. Such a strategy, which depends on the neighborhood of the adversary and builds on a combination of the attacks described in Sections 6.1 and 6.2, will be assumed while deriving the results shown in Section 7.2.

The results, which therefore represent a worst case analysis of the proposed NPV, are shown in terms of the probability that the tests return false positives and false negatives as well as of the probability that a (correct or adversary) node is tagged as unverifiable. In addition, we plot the average difference between the true position of a successful adversary and the fake position it advertises, as well as the overhead introduced by our NPV scheme. The results on attacks aimed at discrediting the position of other nodes are omitted, since they are very close to those we present later in this section.

7.1 Adversaries Attack Strategy

The adversary decision on the kind of attack to launch is driven by the tradeoff between the chances of success and the freedom of choice on its fake position. The **basic** attack allows the adversary to choose any false position, but it requires a high percentage of colluders in the neighborhood in order to be successful. The **hyperbola-based** attack implies less freedom of choice but has higher chances of success. The **collinear** attack pins the adversary into a precise angle with the verifier, and strictly bounds its distance from the verifier itself. However, if the network topology features a sufficient number of collinear nodes, this attack has the highest success probability.

It follows from Section 6 that the best strategy that an adversary can adopt depends on its neighborhood. First, if it colludes with other adversaries outnumbering the noncolluding neighbors, a **basic** attack is launched. Otherwise, if the ratio between colluding and noncolluding neighbors is not greater than (but close enough to) 1, a **hyperbola-based** attack is attempted. As a third option, if noncolluding neighbors greatly outnumber the colluding ones, but some of the former are collinear with the verifier and among themselves, the adversary launches a **collinear** attack. Through it, the adversary can have the noncolluding, collinear neighbors thrown out of the cross-checks in the **CST**. If none of the above conditions are met, the adversary picks a **hyperbola-based** attack, i.e., the one with the

7. An example of suspect situation is the case where the neighborhood of the verifier is split into groups of nodes, whose members pass the cross-checks in the **CST** only with the nodes belonging to the same group.

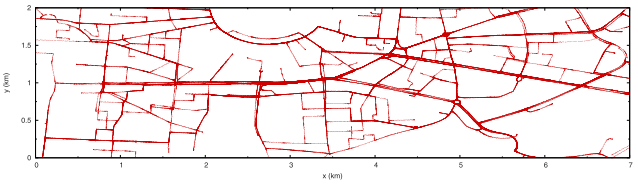


Fig. 8. Road layout of the $7 \times 3\ \mathrm{km}^2$ vehicular scenario.

highest chances of success in absence of noncolluding, collinear neighbors. Also, an adversary always runs a REPLY-**disregard** on *one* noncolluding neighbor, avoiding a mismatch with it. Recall that disregarding just one REPLY does not trigger the **MLT** on the adversary.

7.2 Results

We employed movement traces representing vehicle traffic over a real-world road topology. More precisely, we considered car movements within a $20\ \mathrm{km}^2$ portion of the Karlsruhe urban area depicted in Fig. 8, extracting 3 hours of vehicular mobility that reproduce mild to heavy traffic density conditions. These synthetic traces were generated using the IDM-LC model of the VanetMobiSim simulator, which takes into account car-to-car interactions, traffic lights, stop signs, and lane changes, and has been proven to realistically reproduce vehicular movement patters in urban scenarios [31].

In our simulations, we set $T_{max} = 200\ \mathrm{ms}$, $T_{jitter} = 50\ \mathrm{ms}$, $\Delta = 1\ \mathrm{ms}$ and assume that CSMA/CA is used to access the wireless medium, hence messages can be lost due to collisions. Unless otherwise specified, we fix the proximity range, R, which is equal to the maximum nominal transmission range, to 250 m (resulting in an average neighborhood size of 73.4 nodes), while $\epsilon_r = 6.8\ \mathrm{m}$, $\epsilon_p = 10\ \mathrm{m}$, and the tolerance value $\epsilon_m = 5\ \mathrm{m}$ (roughly corresponding to the case of two vehicles moving at 50 km/h in opposite directions).

To evaluate the performance of our NPV, at every simulation second we randomly select 1 percent of the nodes as verifiers. Then, for each verifier, we compare the outcome of the verification tests with the actual nature of the neighbors. We consider colluding adversaries acting in groups, referred to as *clusters*. Note that a colluding cluster size equal to 1 corresponds to independent attacks. Also, adversaries are knowledgeable, i.e., they perfectly know the identity and location of all colluding and noncolluding neighbors, and always adopt the best attack strategy as described in Section 7.1. In the following, unless otherwise specified, adversaries amount to 5 percent of the overall nodes and are divided into clusters of five colluders each.

In the legend of the plots, **C** stands for correct node (e.g., the label "**C** faulty" refers to the probability of false positives), while **M/Bas**, **M/Hyp**, and **M/Col** stand for adversaries launching, respectively, the **basic**, **hyperbola-based** and **collinear** attack (e.g., the label "**M/Bas** verified" refers to the probability of false negatives due to basic attacks).

We first examine the NPV protocol performance for different values of colluding cluster sizes and $R = 250\ \mathrm{m}$ (Figs. 9a and 9b).

The false negative/positive probability in Fig. 9a clearly shows that 1) the chance of wrong classification reaches 0.01 only for a very large adversarial cluster size, namely 10, 2) the **hyperbola-based** and the **collinear** attacks are the most threatening and 3) an attack by the colluders is most effective in passing themselves off as verified when there are at least three of them. The cluster size also affects the colluders ability to disrupt the positioning of correct nodes, which exhibit as high as a 0.4 percent chance to be tagged as faulty.

Conversely, as shown in Fig. 9b, the cluster size does not cause more correct nodes to be unverifiable, since the main reason for correct nodes to be tagged as unverifiable is the lack of noncollinear neighbors that can verify them. The chance for an adversary to be unverifiable increases with the cluster size, although it is significant only in case of **collinear** attacks. This is in agreement with the fact that the outcome of the **collinear** attack is the avoidance of a sizable number of cross-checks between the adversary and correct nodes, thus likely leading the adversary to be tagged as unverifiable.

The neighborhood size proves to play an important role, as evident in Figs. 9c and 9d where we consider a 5-colluder cluster and vary the transmission range. A small R (hence few neighbors) affects the NPV capability to correctly tag a node. Widening the transmission range with a fixed colluding cluster size significantly favors the verifier, allowing it to reach a conclusive and exact verdict on either correct or adversary nodes: the larger the R, the higher the number of cross-checks involving correct nodes in the **CST**. We note that, for transmission ranges larger than 300 m, we obtain false positive/negative probabilities that are smaller than 0.001. Below 150-m ranges (corresponding to an average neighborhood size of 12 nodes), such probabilities are still 0.01.

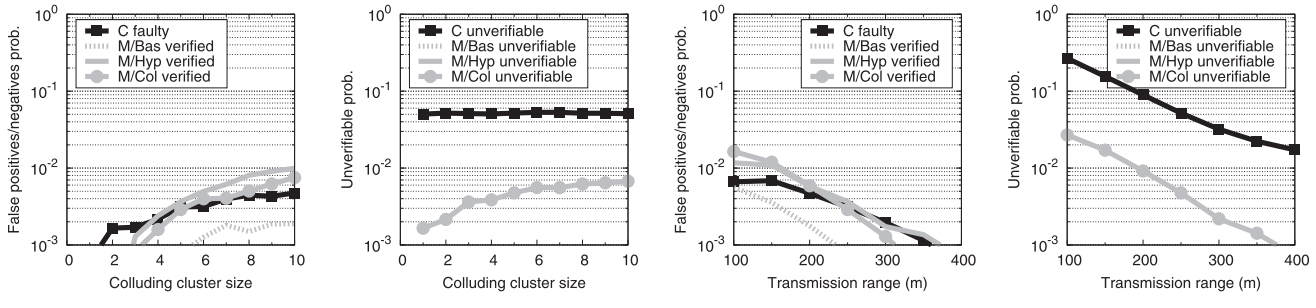


Fig. 9. Probability that a neighbor is tagged incorrectly or as unverifiable, versus the colluder cluster size (a,b), and versus R (c,d). **C**: correct; **M/Bas**, **M/Hyp**, and **M/Col**: adversaries launching the **basic**, **hyperbola-based** and **collinear** attack, each combined with the REPLY-**disregard** attack.

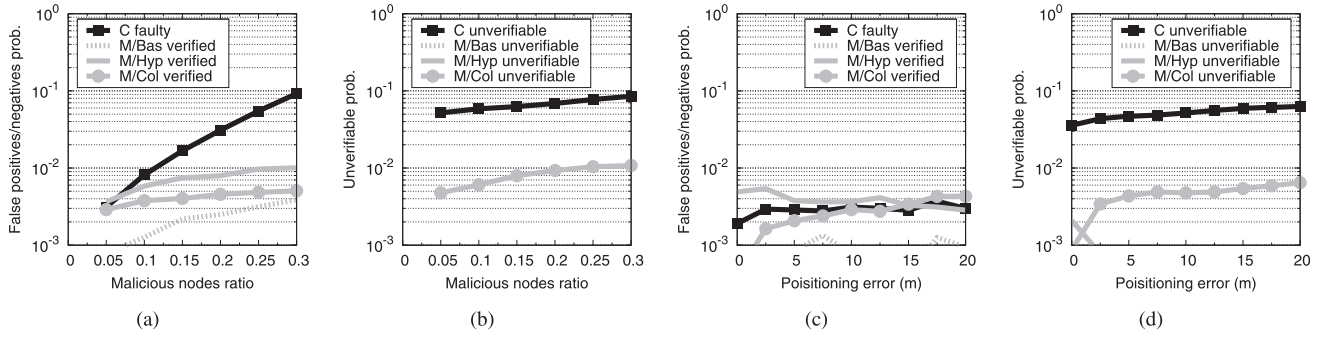


Fig. 10. Probability that a neighbor is tagged incorrectly or as unverifiable, versus the ratio of adversaries (a,b), and position error (c,d). **C**: correct; **M/Bas**, **M/Hyp**, and **M/Col**: adversaries launching the **basic**, **hyperbola-based**, and **collinear** attack, each combined with the REPLY-**disregard** attack.

Beside the impact of the cluster size and of the transmission range, it is important to understand the effect of the percentage of adversaries in the vehicular network. Thus, in Fig. 10a we fix R to 250 m and the cluster size to 5, and we show the robustness of our NPV to the density of adversaries: the probability that adversaries are verified increases ever so slightly with their density. The highest effect is on the probability of correct nodes being tagged as faulty, which however reaches its highest value (0.1) only for 30 percent of adversaries in the network. A further effect of the growing presence of adversaries, as shown in Fig. 10b, is the unverifiable tag being slapped onto more correct nodes. A final observation can be made looking at the false positive/negative probability as the positioning error varies (Figs. 10c and 10d). Interestingly, for any positioning error different from 0, the metrics are only marginally affected.

Finally, we further increase the level of detail of our analysis and study the advantage obtained by adversaries that perform a successful attack against the NPV protocol. Such an adversarial gain is expressed in terms of spatial displacement, i.e., difference of position between the real and fraudulently advertised locations of the successful attacker: clearly, a larger displacement range implies a higher freedom of movement, which, in turn, enables potentially more dangerous actions against the system. The results in Fig. 11a are broken down based on the type of attack launched by the successful adversary, and are limited to the impact of the transmission range, since the other parameters did not show significant influence on the displacement of successful attackers.

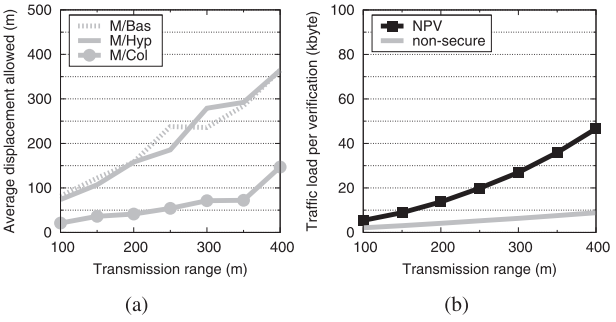


Fig. 11. Displacement gain of adversaries running a successful attack against the NPV (a) and traffic load induced by one instance of the protocol (b).

We can observe that successful **collinear** attacks yield small advantage for adversaries, who are forced to announce positions quite close to their real locations. Moreover, we recall that these attacks constrain adversaries to advertise fake positions along a precise axis, thus further limiting their freedom of movement. We can conclude that **collinear** attacks, typically those with the highest chances of success as previously discussed, are also those resulting in the smallest gain for the adversaries. Conversely, **basic** attacks allow the largest average displacements, but we showed that they have extremely low success probability. The **hyperbola-based** attacks appear then to be the most dangerous ones, if the displacement gain is taken into consideration. However, such a gain becomes significant only for large transmission ranges, in presence of which we already observed that the actual success probability of the attacks becomes negligible.

Finally, we comment on the overhead introduced by our scheme. The NPV protocol generates at most $2n + 2$ messages for one execution initiated by a verifier with n communication neighbors. Also, NPV messages are relatively small in size: with SHA-1 hashing and ECDSA-160 encryption [27], the length of signatures is 21 bytes (with coordinates compression). Assuming that messages include headers with 4-byte source and destination identifiers and 1-byte message type field, POLL, REPLY, and REVEAL are 26, 71, and 67 bytes in size, respectively. The REPORT length depends on the quantity of common neighbor data it carries, amounting to 4 bytes per shared neighbor: information on more than 360 neighbors can thus fit in a single IP packet.

Fig. 11b portrays the traffic induced on the network by one instance of the NPV protocol. The plot only accounts for transmission range variations since, once more, the other parameters do not have an impact on the overhead. We can observe that security comes at a cost, since the traffic load of the NPV protocol is higher than that of a basic nonsecure neighbor position discovery, consisting of only one poll and associated position replies from neighbors. More precisely, the NPV protocol overhead is comparable to that of the nonsecure discovery for smaller transmission ranges, while the difference tends to increase for larger ranges. However, the cost of the NPV protocol is affordable in absolute terms, since one run requires just a few tens of kbytes to be exchanged among nodes, even in presence of dense networks and large transmission ranges. Note that the results above do not take into account the overhead

induced by the distribution of certificates, as it is out of the scope of this work (the interested reader can refer to [26]).

Summary. Given that we assumed the best possible conditions for the adversaries, the above results prove our NPV to be highly resilient to attacks. Indeed, we observed typical probabilities of false positives/negatives below 1 percent, while that of a node being tagged as unverifiable is below 5 percent. Moreover, we showed that a significant portion of the successful attacks yields small advantage to the adversaries in terms of displacement. Finally, the overhead introduced by the NPV protocol is reasonable, as it does not exceed a few tens of kbytes even in the most critical conditions.

8 Conclusion

We presented a distributed solution for NPV, which allows any node in a mobile ad hoc network to verify the position of its communication neighbors without relying on a priori trustworthy nodes. Our analysis showed that our protocol is very robust to attacks by independent as well as colluding adversaries, even when they have perfect knowledge of the neighborhood of the verifier. Simulation results confirm that our solution is effective in identifying nodes advertising false positions, while keeping the probability of false positives low. Only an overwhelming presence of colluding adversaries in the neighborhood of the verifier, or the unlikely presence of fully collinear network topologies, can degrade the effectiveness of our NPV. Future work will aim at integrating the NPV protocol in higher layer protocols, as well as at extending it to a proactive paradigm, useful in presence of applications that need each node to constantly verify the position of its neighbors.

Acknowledgments

This work was supported by the Regione Piemonte through the IoT | ToI project, which is part of the second call of POR F.E.S.R. 2007/2013.

References

[1] *1609.2-2006: IEEE Trial-Use Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages,* IEEE, 2006.

[2] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, "Secure Vehicular Communications: Design and Architecture," *IEEE Comm. Magazine,* vol. 46, no. 11, pp. 100-109, Nov. 2008.

[3] P. Papadimitratos and A. Jovanovic, "GNSS-Based Positioning: Attacks and Countermeasures," *Proc. IEEE Military Comm. Conf. (MILCOM),* Nov. 2008.

[4] L. Lazos and R. Poovendran, "HiRLoc: High-Resolution Robust Localization for Wireless Sensor Networks," *IEEE J. Selected Areas in Comm.,* vol. 24, no. 2, pp. 233-246, Feb. 2006.

[5] R. Poovendran and L. Lazos, "A Graph Theoretic Framework for Preventing the Wormhole Attack," *Wireless Networks,* vol. 13, pp. 27-59, 2007.

[6] S. Zhong, M. Jadliwala, S. Upadhyaya, and C. Qiao, "Towards a Theory of Robust Localization against Malicious Beacon Nodes," *Proc. IEEE INFOCOM,* Apr. 2008.

[7] P. Papadimitratos, M. Poturalski, P. Schaller, P. Lafourcade, D. Basin, S. Čapkun, and J.-P. Hubaux, "Secure Neighborhood Discovery: A Fundamental Element for Mobile Ad Hoc Networks," *IEEE Comm. Magazine,* vol. 46, no. 2, pp. 132-139, Feb. 2008.

[8] Y.-C. Hu, A. Perrig, and D.B. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," *Proc. IEEE INFOCOM,* Apr. 2003.

[9] J. Eriksson, S. Krishnamurthy, and M. Faloutsos, "TrueLink: A Practical Countermeasure to the Wormhole Attack in Wireless Networks," *Proc. IEEE 14th Int'l Conf. Network Protocols (ICNP),* Nov. 2006.

[10] R. Maheshwari, J. Gao, and S. Das, "Detecting Wormhole Attacks in Wireless Networks Using Connectivity Information," *Proc. IEEE INFOCOM,* Apr. 2007.

[11] R. Shokri, M. Poturalski, G. Ravot, P. Papadimitratos, and J.-P. Hubaux, "A Practical Secure Neighbor Verification Protocol for Wireless Sensor Networks," *Proc. Second ACM Conf. Wireless Network Security (WiSec),* Mar. 2009.

[12] M. Poturalski, P. Papadimitratos, and J.-P. Hubaux, "Secure Neighbor Discovery in Wireless Networks: Formal Investigation of Possibility," *Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS),* Mar. 2008.

[13] M. Poturalksi, P. Papadimitratos, and J.-P. Hubaux, "Towards Provable Secure Neighbor Discovery in Wireless Networks," *Proc. Workshop Formal Methods in Security Eng.,* Oct. 2008.

[14] E. Ekici, S. Vural, J. McNair, and D. Al-Abri, "Secure Probabilistic Location Verification in Randomly Deployed Wireless Sensor Networks," *Elsevier Ad Hoc Networks,* vol. 6, no. 2, pp. 195-209, 2008.

[15] J. Chiang, J. Haas, and Y. Hu, "Secure and Precise Location Verification Using Distance Bounding and Simultaneous Multilateration," *Proc. Second ACM Conf. Wireless Network Security (WiSec),* Mar. 2009.

[16] S. Čapkun, K. Rasmussen, M. Cagalj, and M. Srivastava, "Secure Location Verification with Hidden and Mobile Base Stations," *IEEE Trans. Mobile Computing,* vol. 7, no. 4, pp. 470-483, Apr. 2008.

[17] S. Čapkun and J.-P. Hubaux, "Secure Positioning in Wireless Networks," *IEEE J. Selected Areas in Comm.,* vol. 24, no. 2, pp. 221-232, Feb. 2006.

[18] A. Vora and M. Nesterenko, "Secure Location Verification Using Radio Broadcast," *IEEE Trans. Dependable and Secure Computing,* vol. 3, no. 4, pp. 377-385, Oct.-Dec. 2006.

[19] J. Hwang, T. He, and Y. Kim, "Detecting Phantom Nodes in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* May 2007.

[20] T. Leinmüller, C. Maihöfer, E. Schoch, and F. Kargl, "Improved Security in Geographic Ad Hoc Routing through Autonomous Position Verification," *Proc. ACM Third Int'l Workshop Vehicular Ad Hoc Networks (VANET),* Sept. 2006.

[21] J.-H. Song, V. Wong, and V. Leung, "Secure Location Verification for Vehicular Ad-Hoc Networks," *Proc. IEEE Globecom,* Dec. 2008.

[22] M. Fiore, C. Casetti, C.-F. Chiasserini, and P. Papadimitratos, "Secure Neighbor Position Discovery in Vehicular Networks," *Proc. IEEE/IFIP 10th Ann. Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net),* June 2011.

[23] Fed. Highway Administration, "High Accuracy-Nationwide Differential Global Positioning System Test and Analysis: Phase II Report," *FHWA-HRT-05-034,* July 2005.

[24] http://www.nanotron.com/EN/pdf/Factsheet_nanoLOC-NA5TR1.pdf, 2012.

[25] PRECIOSA: Privacy Enabled Capability in Co-Operative Systems and Safety Applications, http://www.preciosa-project.org, 2012.

[26] G. Calandriello, P. Papadimitratos, A. Lioy, and J.-P. Hubaux, "On the Performance of Secure Vehicular Communication Systems," *IEEE Trans. Dependable and Secure Computing,* vol. 8, no. 6, pp. 898-912, Nov./Dec. 2011.

[27] *IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques,* IEEE 1363a 2004, 2004.

[28] M. Ciurana, F. Barcelo-Arroyo, and F. Izquierdo, "A Ranging System with IEEE 802.11 Data Frames," *Proc. IEEE Radio and Wireless Symp.,* Jan. 2007.

[29] F. Carpenter, S. Srikanteswara, and A. Brown, "Software Defined Radio Test Bed for Integrated Communications and Navigation Applications," *Proc. Software Defined Radio Technical Conf.,* Nov. 2004.

[30] E. Del Re, L.S. Ronga, L. Vettori, L. Lo Presti, E. Falletti, and M. Pini, "Software Defined Radio Terminal for Assisted Localization in Emergency Situations," *Proc. First Int'l Conf. Wireless Comm., Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (CTIF Wireless Vitae),* May 2009.

[31] J. Härri, M. Fiore, F. Filali, and C. Bonnet, "Vehicular Mobility Simulation with VanetMobiSim," *Trans. Soc. Modeling & Simulation,* 2009.

Marco Fiore received two MSc degrees from the University of Illinois at Chicago and the Politecnico di Torino in 2003 and 2004, respectively, and the PhD degree from the Politecnico di Torino in 2008. He is an assistant professor at INSA Lyon and an INRIA researcher within the SWING team hosted by the CITI Lab. He has been a visiting researcher at Rice University and the Universitat Politecnica de Catalunya. His research interests are in the field of mobile networking with a focus on vehicular networks. He is a member of the IEEE.

Claudio Ettore Casetti received the graduate degree in electrical engineering from the Politecnico di Torino in 1992 and the PhD degree in electronic engineering from the same institution in 1997. He is an assistant professor in the Dipartime‘nto di Elettronica at the Politecnico di Torino. He has coauthored more than 130 journal and conference papers in the fields of networking and holds three patents. His interests focus on ad hoc wireless networks and vehicular networks. He is a member of the IEEE.

Carla-Fabiana Chiasserini received the PhD degree from Politecnico di Torino in 2000. She has worked as a visiting researcher at UCSD in 1998-2003, and she is currently an associate professor at Politecnico di Torino. Her research interests include architectures, protocols, and performance analysis of wireless networks. She has published more than 190 papers in prestigious journals and leading international conferences, and she serves as an associated editor of several journals. She is a senior member of the IEEE and the IEEE Computer Society.

Panagiotis Papadimitratos received the PhD degree from Cornell University, Ithaca, New York. He was a postdoctoral fellow at Virginia Tech, Blacksburg, VA, a scientist at EPFL, Lausanne, Switzerland, and a visiting faculty member at Politecnico di Torino. He is now an associate professor in the School of Electrical Engineering at KTH, Stockholm, Sweden. His research is concerned with security and networked systems, with more than 70 related technical publications. He is a member of the IEEE.