Multi-Hop Distance Estimation: How Far are You?

Aikaterini Mitrokotsa
aikmitr@chalmers.se

Cristina Onete,
cristina.onete@gmail.com

Elena Pagnin
elenap@chalmers.se,

Mahesh Perera
maheshp@student.chalmers.se

July 17, 2017

Abstract

Several access control systems are based on the users' physical location/proximity to the access point. Distance-Bounding (DB) protocols constitute a classical solution to calculate the distance between a trusted verifier (e.g., an access point) and an untrusted prover (e.g., a pervasive device). The main limitation of DB is that the prover and the verifier need to lie in each other's communication range. In this paper, we introduce the concept of Multi-Hop Distance-Estimation (MHDE) protocols, which enable a verifier to authenticate a possibly far-away prover and estimate its distance to this prover, when they are not in the communication range of each other, using an ad-hoc network of pervasive devices. More precisely, our contributions are three-fold, since we provide: (1) a formal definition for MHDE; (2) a threat model for MHDE that considers a powerful and distributed adversary; and (3) implementation of MHDE protocols with different settings. Additionally, we demonstrate our protocol to be secure in the considered threat model, and we provide a performance analysis regarding the accuracy of the distance estimation and the tolerance of limited mobility of the nodes. The results are promising in order to adopt MHDE in a distributed setting.

1 Introduction

A new concept in computer science is that of *ubiquitous computing*. It models the fact that modern-day users, equipped with mobile devices of various potence, can process information at any time and place. Mobility has extended from using laptops to the use of tablets, terminals, smartphones, and more. Two fundamental requirements of ubiquitous computing, on which most applications rely on, are: authentication (in particular device-to-device authentication) and relative-positioning (such as the distance between devices or network topology). In this paper, we merge these two concepts into a new notion which enables authentication *and* distance-estimation across a network of ubiquitous devices: *multi-hop distance-estimation* (MHDE).

More concretely, *authentication* protocols enable a *prover*, $\mathcal{U}$, to prove its legitimacy to a *verifier*, $\mathcal{V}$, over a *direct*, insecure channel. While preserving the same functionality, our MHDE proposal provides authentication in case the prover, $\mathcal{U}$, and the verifier, $\mathcal{V}$, are not in each other's proximity, but rather are part of an ad-hoc network of pervasive devices, e.g., sensor networks, Vehicle Ad-Hoc Networks (VANETs) or, more generally, Mobile Ad-Hoc Networks (MANETs). A secondary goal, apart from *authenticating* provers over multiple hops, is to *approximate* their distance to an honest verifier.

There are several motivating application scenarios for MHDE protocols. Consider, for instance, a parking payment system [17]. Distance estimation would provide a new method to compute discounts (or lower rates) to cars parked within lower-interest zones (and far from points of interest such as a mall). In this case, a verifier within the low-interest zone will be able to approximate its distance to cars parked in the area (which are equipped with pervasive devices able to connect in ad-hoc networks), and it will separate those parked nearby from those parked far. Another example is a zoo where the animals have embedded chips (e.g., RFID tags or more powerful devices). The resulting network could allow the staff to check whether dangerous animals (acting as provers) are still within their reserved area, where a special device (access point, acting as the verifier) is located, or whether the animals have escaped. Similarly, a prisoner who is released on parole and is allowed to live within society without leaving a restricted area, could benefit of authenticated distance-estimation. The parole officer can keep track of the prisoner's whereabouts via MHDE, under the assumption that the prisoner will at all times wear a physically unremovable device which is always connected and will perform regular authentication. In this scenario, MHDE can rely on the ad-hoc network naturally formed by pervasive devices populating the prisoner's neighbourhood to authenticate him and approximate his distance to the parole officer's building (verifier).

Contributions. In this paper, we define multi-hop distance-estimation (MHDE), which is designed for networks formed by pervasive devices and aims at enabling a verifier to authenticate a prover, while estimating the distance to one another. We use a provable-security approach and we support our claims with experimental evaluations. We formally define the syntax of MHDE protocols as well as a threat model. We propose the first MHDE protocol and prove its security in the considered threat model. In our construction, we make use of well-studied cryptographic primitives such as: public-key

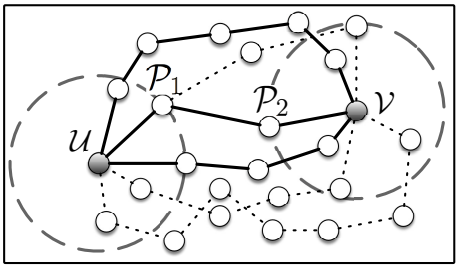


Figure 1: Communication network for multi-hop distance-estimation (MHDE). The two dashed circles delimit the communication range δ of the nodes. Bold edges indicate possible paths chosen by $\mathcal{V}$ to estimate its distance from $\mathcal{U}$.

encryption, existentially unforgeable signatures, and computationally hiding and binding commitments. In addition, we perform some evaluations to analyse the reliability and the accuracy (in terms of distance-estimation-error DEE[1]) of our scheme when three different metrics are used to estimate the distance and when the nodes in the network are allowed to have limited mobility. Our measurements show that the ratio between the estimated distance and the real one decreases sub-linearly in the length of the chosen paths.

2 Related Work

Estimating the location of a node in a communication network, as well as estimating its distance from a specific spot (e.g., access point), or defining the network topology, are areas that have received a lot of attention in recent years. The notion that lies closest to MHDE is the one of Distance-Bounding [3]. However, our aim in designing MHDE is different. In MHDE instead of just verifying that two parties are *close enough*, we use a network to estimate *how far* two nodes are. Moreover, distance-bounding assumes that the prover and verifier are in direct communication, whereas we assume the existence of many network nodes between these two end points. We nevertheless use a construction akin to the protocol of Brands and Chaum [3] as a base building block to our scheme.

Other research areas connected to distance estimation (but quite different) are Secure Neighbour Discovery (SND) [14], secure localisation [4,11,12] and position-based cryptography [5]. Contrary to the majority of localisation schemes, in this work we avoid the use of Global Positioning Systems (GPS), which are known to be vulnerable to spoofing attacks and require special hardware and additional infrastructure [15]. While existing localisation and neighbour-discovery approaches rely on specialised hardware [13], require an external trustworthy infrastructure, or offer limited security guarantees, our MHDE approach aims at avoiding these.

3 Preliminaries

Throughout the paper, $\mathcal{N}$ denotes a large, dynamic, noiseless network. The nodes of $\mathcal{N}$ are called users and will be denoted by calligraphic letters of the alphabet, e.g., $\mathcal{P}$. These nodes can be classified in three categories: the *prover* $\mathcal{U}$; the *verifier* $\mathcal{V}$; and the *proxies* $\mathcal{P}_i$. The prover is a special user that aims to authenticate to another special node, the verifier, and it can be honest or malicious. The verifier is an honest node, which: (1) authenticates any legitimate prover $\mathcal{U}$; and (2) approximates the distance between the prover and the verifier. The *proxies* $\mathcal{P}_i$ are the remaining nodes in $\mathcal{N}$ and may be either honest or malicious. Each node is associated with a *broadcast range*, corresponding to the signal-power radius of the device. We assume that *honest* users out of the broadcasting range of a node $\mathcal{P}$ cannot directly *see* any messages broadcasted by that user. In our security model, we assume all users have the same range $\delta > 0$; this simplifies our analysis. On the other hand, in our implementations we also investigate how the protocol behaves if the broadcasting ranges are not constant.

As a matter of notation, for any $n \in \mathbb{N}$, we indicate by $[n]$ the set $[n] := \{1, \ldots, n\}$. Figure 1 provides a visual example of the networks we consider in this paper. Formally, we model the communication network $\mathcal{N}$ as a graph $\{\mathbf{V}, \mathbf{E}\}$, where the set of vertices $\mathbf{V} = \{\mathcal{P}_i \ : \ i \in [N+2]\}$ is the set of nodes, including the proxies and the two special nodes $\mathcal{U} = \mathcal{P}_{N+1}$ and $\mathcal{V} = \mathcal{P}_{N+2}$. Edges between users signify that the two devices are in each other's broadcasting ranges, i.e. they are *connected*. The set of edges $\mathbf{E}$ may change over time as users move; however, we assume that the graph is always connected and each node has degree at least $3k$, where k is the maximal number of *untrusted* nodes in $\mathcal{N}$ (see Section 5 for details).

In this paper, we consider a metric for the physical distance between two nodes at a given time, denoted by $\mathsf{dist}(\cdot, \cdot)$; this is usually the Euclidean distance in the 3-dimensional space.

[1] Our distance-measurements use several *paths* to estimate the distance between two nodes; this will introduce an error in our measurements, which we amply investigate through experimentation.

4 High-level Overview of MHDE

In this section, we present a high level overview of our *Multi-Hop Distance-Estimation* protocol (MHDE in short). Intuitively, MHDE runs among nodes (registered) in a communication network $\mathcal{N}$. The protocol is initiated by the verifier $\mathcal{V}$, who wants to *find* a target node, the prover $\mathcal{U}$, within the network. To do so, MHDE uses the network nodes (proxies) that lie between the prover and the verifier just like milestones, in order to estimate the distance $\Delta_{\mathcal{U},\mathcal{V}}$ between $\mathcal{U}$ and $\mathcal{V}$. For correctness, we also require that the verifier can determine a value $\Delta_{\mathcal{U},\mathcal{V}}$ that estimates the distance between $\mathcal{V}$ and $\mathcal{U}$, and not another node impersonating $\mathcal{U}$.

At a high-level, an MHDE protocol is a collection of interactive algorithms with different objectives. We explain here the main phases of the protocol flow. We refer the reader to Appendix .2 for formal descriptions.

Set-up: The scheme begins with a set-up, where some global (public) parameters ppar are generated. Users can *register* to the network as proxies; thus generating user secret parameters and public parameters. The latter are included in the ppar. For each desired prover-verifier pair, the protocol generates dedicated keys, in particular a public-secret key pair for $\mathcal{V}$, and a secret key K shared (only) between $\mathcal{U}$ and $\mathcal{V}$.

Paths: To estimate the distance of $\mathcal{U}$ from $\mathcal{V}$, the verifier runs a path-generation algorithm which outputs a list of paths from the prover to the verifier. In general, this algorithm relies on the routing protocol used in the network, we show an alternative solution in Appendix .1.

Distance Estimation: This is the core part of the MHDE protocol. It starts by $\mathcal{V}$ selecting a number of paths between $\mathcal{U}$ and $\mathcal{V}$ (identified in the previous phase). Then, for each path, the proxies that compose that path run a time-critical challenge-response protocol. This consists of a suite of interactive algorithms that should enable the verifier to measure the challenge-response round trip times of the messages sent from $\mathcal{U}$ to $\mathcal{V}$ via the in-between proxies.

Conclusion: At this point of the MHDE protocol, the verifier has collected enough data to produce an estimation $\Delta_{\mathcal{U},\mathcal{V}}$ of the distance between $\mathcal{V}$ and $\mathcal{U}$. To achieve authentication, however, we need to run an additional suite of interactive algorithms. Intuitively, we ask the prover to disclose some pieces of information, that enable the verifier to identify $\mathcal{U}$ correctly. However, since the prover and the verifier are outside of each other's communication range, these messages need to travel through the nodes in the paths that connect $\mathcal{U}$ to $\mathcal{V}$. This last condition is what makes this step interactive.

Our MHDE protocol has a two-data output: (1) an authentication bit $\mathsf{b} \in \{0,1\}$ (b=1 if the prover is authenticated), and (2) an estimation $\Delta_{\mathcal{U},\mathcal{V}}$ of the physical distance $\mathsf{dist}(\mathcal{U},\mathcal{V})$ between $\mathcal{U}$ and $\mathcal{V}$.

We observe that our MHDE protocol may return slightly different outputs when run between a prover and a verifier placed at the same distance. This is due, for instance, to the network topology and the number of proxies in the chosen paths. Note also that the transcripts of the protocol may also differ between two executions with the same network setup, and the paths chosen by $\mathcal{V}$.

The following definition extends the usual notion of correctness in authentication to include the concept of distance-estimation.

Definition 1 ($(\epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$-Correctness)**.** *A multi-hop distance-estimation protocol* MHDE *is* $(\epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$*-correct if, for all possible output of honestly-run algorithms in the protocol, the final output* $(\mathsf{b}, \Delta_{\mathcal{U},\mathcal{V}})$ *satisfies*

- ***Authentication:*** $\mathsf{Prob}\left[\mathsf{b}=0\right] < \epsilon_{\mathsf{auth}}$;
- ***Distance-estimation:*** $\mathsf{Prob}\left[|\Delta_{\mathcal{U},\mathcal{V}} - \mathsf{dist}(\mathcal{U},\mathcal{V})| \geqslant \delta_{\mathsf{dist}}\right] < \epsilon_{\mathsf{dist}}$.

In Section 8, we analyse the deviation of the value $\Delta_{\mathcal{U},\mathcal{V}}$ estimated by the MHDE protocol from the actual Euclidean distance between the prover and the verifier $\mathsf{dist}(\mathcal{U},\mathcal{V})$. This study allows to gain information about the *minimal* value δ_{dist} to guarantee the correctness of the protocol.

5 Adversarial Model

For MHDE we consider a malicious adversary $\mathcal{A}$ that can choose the topology of the network and adaptively corrupt users, i.e., learn their secret key and behave dishonestly. We denote the set of adversarial-controlled nodes by $\mathbf{P}^* \subset \mathcal{N}$. We assume that dishonest parties are able to communicate between one another independently of the physical distance, to broadcast messages (as they would if they were honest) and to uni-cast messages to any other user in the network at their will. In addition, $\mathcal{A}$ may collect all the messages that are interchanged in the network.

In our security analysis, the verifier $\mathcal{V}$ is always honest; the prover $\mathcal{U}$ may be malicious (in which case it is denoted by $\mathcal{U}^*$), and we set a upper-bound k on the malicious parties present in the system. In particular, in all attacks, we let $|\mathbf{P}^*| \leqslant k$, apart from the distance shortening attach, in which $\mathbf{P}^* = \mathcal{U}^*$.

5.1 Security Notions

In distance-estimation authentication protocols, the verifier has two goals: (1) to authenticate the prover; and (2) to estimate its distance to the prover. Consequently, the adversary may aim to thwart either one or both of these goals.

We consider three types of attacks, which resemble *mafia fraud* (MiM-Attacks), *distance fraud* (Distance-Shortening), and *terrorist fraud* (Collusion-Attacks) in distance-bounding authentication [1, 2, 6]. Both our security analysis and our simulations assume that $\mathcal{A}$ controls the highest admissible number of (malicious) nodes.

MiM-Attacks. In this scenario, both the prover, $\mathcal{U}$, and the verifier, $\mathcal{V}$, are honest. The adversary $\mathcal{A}$ plays the following *game*: The adversary chooses a network topology and the nodes are registered according to it. The MHDE protocol is started until the end of the *path* phase. Meanwhile $\mathcal{A}$ can adaptively select a set $\mathbf{P}^* \subset \mathcal{N} \setminus \{\mathcal{U}, \mathcal{V}\}, |\mathbf{P}^*| \leqslant k$ of malicious proxies (which represent multiple Men-in-the-Middle entities). The corruption phase ends when the time-critical phase starts. The MHDE protocol proceeds until its end, and the final outputs are given to $\mathcal{A}$. The adversary is allowed to have q sessions, i.e., $\mathcal{A}$ may run again the game from the beginning of the time-critical phase. In each session, the set of paths chosen by $\mathcal{V}$ are the same (since this is set the first time the game runs, and no longer updated). Also, all nodes in the network have the same keys. What variates is the randomness chosen by the party involved in the game.

We distinguish two MiM-attack scenarios: impersonating the prover (MiM-Impersonation) and/or tampering with the verifier's estimation of the distance to the prover (MiM-Distance-Perversion). Intuitively, the winning conditions are: for *MiM-Impersonation*, there exists a protocol session in which the verifier's authentication bit is $\mathsf{b} = 1$ and $\mathcal{U}$ did not take part; for *MiM-Distance-Perversion*, there exists a protocol session in which the honest $\mathcal{U}$ participated, but the distance estimated by the verifier in the end is larger than the real one (by more than δ_{dist}, the scheme's accuracy value). More formally:

Definition 2 ($(q, \epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$-MiM -Resistance)**.** *A $(\epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$-correct multi-hop distance-estimation authentication protocol* MHDE *is MiM-Resistant if, for all adversaries $\mathcal{A}$ playing the game outlined above and making at most q protocol sessions π_i with corresponding outcomes $(\mathsf{b}^{(i)}, \Delta^{(i)}_{\mathcal{U},\mathcal{V}})$, it holds that: if $\mathsf{Prob}\left[\exists i \in [q] : \mathsf{b}^{(i)} = 1\right] \geqslant \epsilon_{\mathsf{auth}}$, for a session π_i in which both the prover and the verifier were honest, then one of the following conditions is satisfied:*

- ***MiM-Impersonation**: The prover $\mathcal{U}$ did not participate in session π_i; or*
- ***MiM-Distance-Perversion**: The prover $\mathcal{U}$ initiated session π_i and the final distance estimation value $\Delta^{(i)}_{\mathcal{U},\mathcal{V}}$ output by the verifier at the end of session is such that: $\mathsf{Prob}\left[\ |\mathsf{dist}(\mathcal{U}, \mathcal{V}) - \Delta_{\mathcal{U},\mathcal{V}}| \geqslant \delta^{(i)}_{\mathsf{dist}}\right] \geqslant 1 - \epsilon_{\mathsf{dist}}$.*

Distance-Shortening. In this attack, $\mathbf{P}^* = \{\mathcal{U}^*\}$ and the adversary's goal is to make the verifier estimate a shorter distance than the real one between $\mathcal{U}^*$ and $\mathcal{V}$. The adversary $\mathcal{A}$ plays the following *game*: The adversary chooses a network topology, in particular the location of the prover. The nodes are registered according to this topology and $\mathcal{A}$ immediately corrupts $\mathcal{U}$. The MHDE protocol is then run in this setting and the final outputs are given to $\mathcal{A}$. The adversary is allowed to have q protocol sessions in this game, with the same setting initial setting (i.e. users' keys and paths chosen by $\mathcal{V}$).

The adversary wins the game if there exists at least one protocol session in which the estimated $\Delta_{\mathcal{U}^*,\mathcal{V}} < \mathsf{dist}(\mathcal{U}^*, \mathcal{V})$. More formally:

Definition 3 ($(q, \epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$-DS-Resistance)**.** *A $(\epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$-correct multi-hop distance-estimation authentication protocol* MHDE *is Distance-Shortening-resistant if, for all adversaries $\mathcal{A}$ playing the game outlined above and running at most q protocol sessions, the final output of the protocol is ($\mathsf{b} = 1$, $\Delta_{\mathcal{U}^*,\mathcal{V}}$), and it holds that:*
$\mathsf{Prob}\left[\Delta_{\mathcal{U}^*,\mathcal{V}} < \mathsf{dist}(\mathcal{U}^*, \mathcal{V}) - \delta_{\mathsf{dist}}\right] < \epsilon_{\mathsf{dist}}$.

Collusion-Attack. Collusion attacks are particularly insidious, since they involve the cooperation between the network adversary and a malicious prover. Consider a parking lot, in which employees of a company may park their car in parking places that are close to their main buildings (but not around other buildings). A verifier placed within the center of the parking lot will estimate distances between the potential parked cars and itself, by using other cars as intermediate nodes. Suppose that, for one particular day, a legitimate prover might decide to allow a friend to park in their spot by allowing that friend to impersonate it. However, the prover would not want its friend to continue using its parking spot.

More formally, the dishonest prover $\mathcal{U}^*$ *collaborates* with the adversary to make the verifier miss-estimate $\mathsf{dist}(\mathcal{U}^*, \mathcal{V})$, while being authenticated (i.e., $\mathsf{b} = 1$) and without leaking its secret keys to $\mathcal{A}$. Our model of collusion attacks resembles the terrorist-fraud attack of [1, 9, 16] in the context of DB.

We define the following *2-phase game* for collusion attacks. In the *first phase*, the Malicious-Prover phase (MP), the adversary chooses a network topology and the nodes are registered according to it. The MHDE protocol proceeds till the end of the *path* phase. Meanwhile $\mathcal{A}$ can adaptively define the set $\mathbf{P}^* \subset \mathcal{N} \setminus \{\mathcal{U}\}, |\mathbf{P}^*| \leqslant k$. The MHDE protocol proceeds until its end, and the final outputs are given to $\mathcal{A}$. As in the other games, $\mathcal{A}$ may run q protocol sessions in this setting. Note that since $\mathcal{U}^*$ is malicious it may not behave honestly in this phase. In this phase the aim of $\mathcal{U}^*$ and $\mathcal{A}$ is to: (1) authenticate to the verifier; and (2) ensure that the estimated distance $\Delta_{\mathcal{U}^*,\mathcal{V}}$ does not differ of more than δ_{dist} from the actual one.

The *second phase* is called Honest-Prover phase (HP). The network has the same setting as in the previous phase. The adversary plays an MiM-Impersonation game against the prover, who is not taking part to the protocol. Again, $\mathcal{A}$ is allowed to have up to q protocol sessions in this setting. We say that the MP-phase failed ($\mathcal{A}$ loses the MHDE collusion game) if in the HP-phase $\mathcal{A}$ successfully authenticates to the verifier. Formally:

Definition 4 ((q, ϵ_{auth})-failed MP-phase)**.** *The adversary $\mathcal{A}$ (defined above) has* failed *its MP-phase if either one of the following conditions holds:*

- *For each of the MP sessions π_i^{MP}, $i \in [q]$, it holds that: either the verifier's final output contains $\mathsf{b}^{(i)} = 0$ with probability ϵ_{auth}; or the output is $(1, \Delta_{\mathcal{U}*,\mathcal{V}})$ and $\mathsf{Prob}\left[|\mathsf{dist}(\mathcal{U},\mathcal{V}) - \Delta_{\mathcal{U}*,\mathcal{V}}| \geqslant \delta_{\mathsf{dist}}^{(i)}\right] \geqslant 1 - \epsilon_{\mathsf{dist}}$.*
- *There exists an HP-session in which $\mathcal{U}$ did not take part, and the verifier's output contains $\mathsf{b} = 1$, i.e., $\mathsf{Prob}\left[\exists i \in [q] : \pi_{i*}^{\text{HP}} \; and \; \mathsf{b}^{(i)} = 1\right] \geqslant \epsilon_{\mathsf{auth}}$.*

We define *collusion resistance* as follows:

Definition 5 ((q,p,ϵ_{auth})-Collusion-Resistance)**.** *An* MHDE *authentication protocol is $(q, \epsilon_{\mathsf{auth}})$-Collusion-resistant if, for all sets of dishonest entities $\{\mathcal{U}^*, \mathbf{P}^*\}$ playing an MP-collusion attack with q sessions and winning with non-negligible probability p, there exists an HP phase adversary $\mathcal{A} \subseteq \mathbf{P}^*$ with respect to which the proxies $\{\mathcal{U}^*, \mathbf{P}^*\}$ ϵ_{auth}-failed the MP-phase.*

We stress that in the collusion attack we consider both distance shortening and distance enlarging. In the latter case, the number of hops in the (malicious) paths between $\mathcal{U}$ and $\mathcal{V}$ already upper bounds the expected round-trip time, as we discuss in Appendix .5; thus $\mathcal{A}$ cannot enlarge the distance *too much*. By using implementation results, we indicate some possible values for the value δ_{dist} in Section 8.

6 The Protocol

Notations. The users in the network $\mathcal{N}$ within the broadcasting range of any node $\mathcal{P}$ form the set denoted $\mathcal{N}[\mathcal{P}]$ (the neighbours of $\mathcal{P}$). Unless differently specified, the paths we consider in the network are from $\mathcal{U}$ to $\mathcal{V}$; they are denoted as a collection of nodes (without the edges between them), e.g., $\mathsf{Path}_{(1)} = \{\mathcal{U}, \mathcal{P}_1^{(1)}, \mathcal{P}_2^{(1)}, \ldots, \mathcal{P}_{\mathsf{len}[1]}^{(1)}, \mathcal{V}\}$, where $\mathsf{len}[j]$ indicates the *length* of $\mathsf{Path}_{(j)}$. To each path we append a publicly verifiable proof $\Pi_{(j)}$ of: (1) the order of nodes, and (2) the fact that each user in the indicated list of nodes is actually on the path. Two paths are said to be *disjoint* if the only common nodes are $\mathcal{U}$ and $\mathcal{V}$ (the *root* and the *leaf* respectively).

To improve readability, we defer the nitty-gritty description of our MHDE protocol to Appendix .3 and present here the behaviour of the main algorithms.

6.1 Protocol's Preamble

In the protocol's *preamble* we make use of a Trusted Third Party (TTP) to run the network initialisation algorithms; however, as soon as the network is set up, we no longer need the TTP (it does not contribute to the protocol run).

$\underline{\mathsf{Setup}}(\cdot)$: The setup consists of a TTP generating the parameters for a computationally hiding and binding commitment scheme $\mathsf{CScheme} = (\mathsf{Commit}, \mathsf{Open})$, a public-key encryption scheme $\mathsf{EScheme} = (\mathsf{EKGen}, \mathsf{Enc}, \mathsf{Dec})$, and an unforgeable signature scheme $\mathsf{SScheme} = (\mathsf{SKGen}, \mathsf{Sign}, \mathsf{Vf})$.

$\underline{\mathsf{UserRegistration}}(\cdot,\cdot)$: This algorithm requires a user *identity* and the user's *role* (prover, verifier, common proxy); its role is to register the user with the TTP. The TTP stores in a database: the user identity $\mathcal{P}$ and the registered parameters. No double registrations is allowed. The parameters output by the TTP for user $\mathcal{P}$ are: (1) a tuple of secret/public signature parameters $(\mathsf{ssk}, \mathsf{spk})$ generated by SKGen; (2) a random n-bit string K. The secret parameters are transmitted securely, while the public parameters are made available to all parties. If a node registers as *verifier*, the TTP additionally generates a key-pair for the PKE scheme: $(\mathsf{epk}, \mathsf{esk}) \leftarrow \mathsf{EKGen}()$. The public key epk is made public, while esk and the strings K of all registered users are given to the verifier solely.

$\underline{\mathsf{Paths}}(\cdot)$: As a setup assumption we state the existence of an algorithm, $\mathsf{PathGen}$, which given any node in the network returns a list of paths from that node to the verifier. The paths output by $\mathsf{PathGen}$ are validated, i.e., the verifier removes from the list any path that does not have the structure $\left(\{\mathcal{U}, \mathcal{P}_1^{(j)}, \ldots, \mathcal{P}_{\mathsf{len}[j]}^{(j)}, \mathcal{V}\}, \Pi_{(j)}\right)$ above, or for which the proof $\Pi_{(j)}$ is incorrectAfter validating the path list, $\mathcal{V}$ chooses a set $\mathcal{D}$ of $2k+1$ (apparently) disjoint paths of length between $\mathsf{MinLength}$ and $\mathsf{MinLength} + \nu$, where $\mathsf{MinLength}$ denotes the minimum path length and ν is a tolerance rate (in number of hops)[2]. The verifier may run $\mathsf{PathGen}$ multiple times until there are $2k+1$ paths satisfying these properties. The reason why the verifier can choose paths that are only *apparently* rather than *effectively* disjoint lies in the fact that $\mathcal{A}$ may mount wormhole attacks using the nodes in $\mathbf{P}^*$. Figure 2 depicts an explicating example of apparently disjoint paths, which in reality connect to just one crucial malicious proxy, $\mathcal{P}_1^*$. The set $\mathcal{D}$ is not uniquely determined; however, this does not harm the security of our MHDE protocol. In Section 8, we show how the choice of $2k+1$ affects the distance-estimation error.

[2]This selection of paths minimizes the distance-estimation error, since we exclude paths that have many more nodes than the minimal path. Scenario-specific simulation should provide an optimal choice of ν depending on the network density and the desired accuracy δ_{dist}.

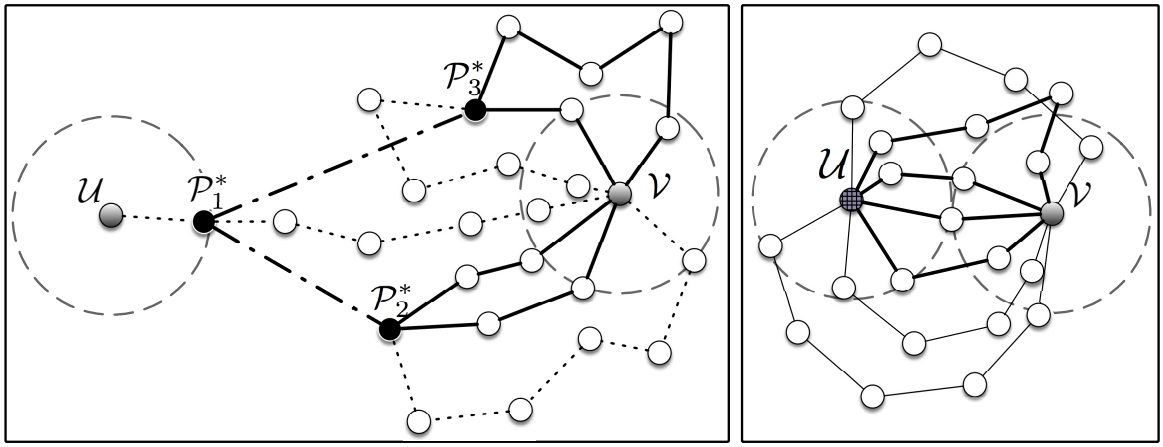


Figure 2: Wormhole attack. On the left hand side, the real network. On the right hand side, the topology guessed by $\mathcal{V}$. Each malicious node $\mathcal{P}_1^*, \mathcal{P}_2^*, \mathcal{P}_3^*$ is claiming to be $\mathcal{U}$, relaying through a *wormhole* the prover's messages.

6.2 Protocol's core

The intuition behind core part of our MHDE protocol is quite simple: path authentication is achieved using digital signatures and distance estimation is based on round-trip timings. Nonetheless, a careful use of cryptographic primitives is needed in order to reach the high level of security we aim for. In MHDE the verifier is equipped with a clock, and we separate communication rounds from the verifier to the prover (through the proxies) into two categories: time-critical (if the verifier measures the round-trip time between sending a message and receiving – through various hops – the response) and slow (if no clock is used).

Figure 3 depicts the protocol flow in this core part. For each path $\mathsf{Path}_{(j)} \in \mathcal{D}$, the following steps are run subsequently:

Nominate$(\cdot)$: The path nomination algorithm takes as input the set $\mathcal{D}$ determined in the previous phase, it signs each path $\mathsf{Path}_{(j)} \in \mathcal{D}$ and broadcasts the the signatures to the nodes in the path. This algorithm is run by the verifier solely.

MHInit: This is a collection of interactive algorithms run among the proxies in $\mathsf{Path}_{(j)}$. Upon receiving the broadcast message output by Nominate, each proxy $\mathcal{P}_\ell^{(j)}$ on $\mathsf{Path}_{(j)}$ selects uniformly at random an n-bit string called $\mathsf{offset}[\mathcal{P}_\ell^{(j)}]$. The offset is committed to a value $\omega_\ell^{(j)}$, which is then signed (using the proxy's signing key $\mathsf{ssk}_{\mathcal{P}^{(j)}}$). Let $\sigma_\ell^{(j)}$ denote the signature over $\omega_\ell^{(j)}$. The pair $(\omega_\ell^{(j)}, \sigma_\ell^{(j)})$ is forwarded to the *next node* along the path. Each proxy relays the data received by its neighbour until all pairs commitment-signature reach $\mathcal{V}$. The verifier checks that the signatures are correct *and* the order in which they were received is consistent with the structure of the path. If any condition is not satisfied, $\mathcal{V}$ nominates a different path[3]. Note that the adversary can use this ample verification phase to cause a denial-of-service attack. However, in this paper, DoS attacks are out of scope.

MHDistEst: In our instantiation, distance-estimation is a suite of n time-critical exchanges, each having the same structure. For every $\mathsf{Path}_{(j)}$, $\mathcal{V}$ sends a message along the path to $\mathcal{U}$, and a response follows the reverse way:

$$\mathcal{V} \to \mathcal{P}_{\mathsf{len}[j]}^{(j)} \to \ldots \to \mathcal{P}_1^{(j)} \to \mathcal{U} \to \mathcal{P}_1^{(j)} \to \ldots \to \mathcal{P}_{\mathsf{len}[j]}^{(j)} \to \mathcal{V}$$

Concretely, $\mathcal{V}$ picks a 1-bit random challenge $c_i \in \{0,1\}$, it starts its clock, and it sends c_i to the first node in the selected path $\mathsf{Path}_{(j)}$. Whenever a proxy (distinct from $\mathcal{V}, \mathcal{U}$) receives a 1-bit value $\hat{c}_i$, it forwards it to the next node (in reverse order) along the path, until the message reaches $\mathcal{U}$. The prover, $\mathcal{U}$, computes the response $r_i^{(j,0)} = \mathsf{offset}[\mathcal{U}]_i \oplus (c_i \cdot \mathsf{K}_i)$, using the i-th bit of the secret key K that is shared between $\mathcal{U}$ and $\mathcal{V}$. Then, $\mathcal{U}$ sends this response to the next node (this time in the correct order) in the path $\mathsf{Path}_{(j)}$. Each proxy $\mathcal{P}_\ell^{(j)} \notin \{\mathcal{U}, \mathcal{V}\}$ receives from its neighbour a 1-bit response $r_i^{(j,\ell-1)}$ and XORs this value with its offset, computing $r_i^{(j,\ell)}$. Finally, the last response reaches $\mathcal{V}$, who stops the clock upon receiving it. The verifier stores the received response value together with the time $\Delta t_i^{(j)}$ recorded from the clock for that round.

MHVerification: In the final phase of the MHDE protocol each node in the path – other than $\mathcal{V}$ – generates values to authenticate itself. The verifier, instead, checks if these values are correct and coherent with the protocol's transcript in order to authenticate the prover and estimate its distance. More precisely, $\mathcal{U}$ opens its commitment and *encrypts* the opening using the verifier's epk. Encryption is required in order to prevent key-recovery attacks à la [8, 10] against the prover's secret key K. Finally, $\mathcal{U}$ signs both the encryption of the commitment and the protocol's transcript from its point of view and sends the ciphertext and the signature to the next node. Each proxy in the path opens the commitment on its

[3] This is to avoid to use a path with un-identified nodes in the distance estimation phase.

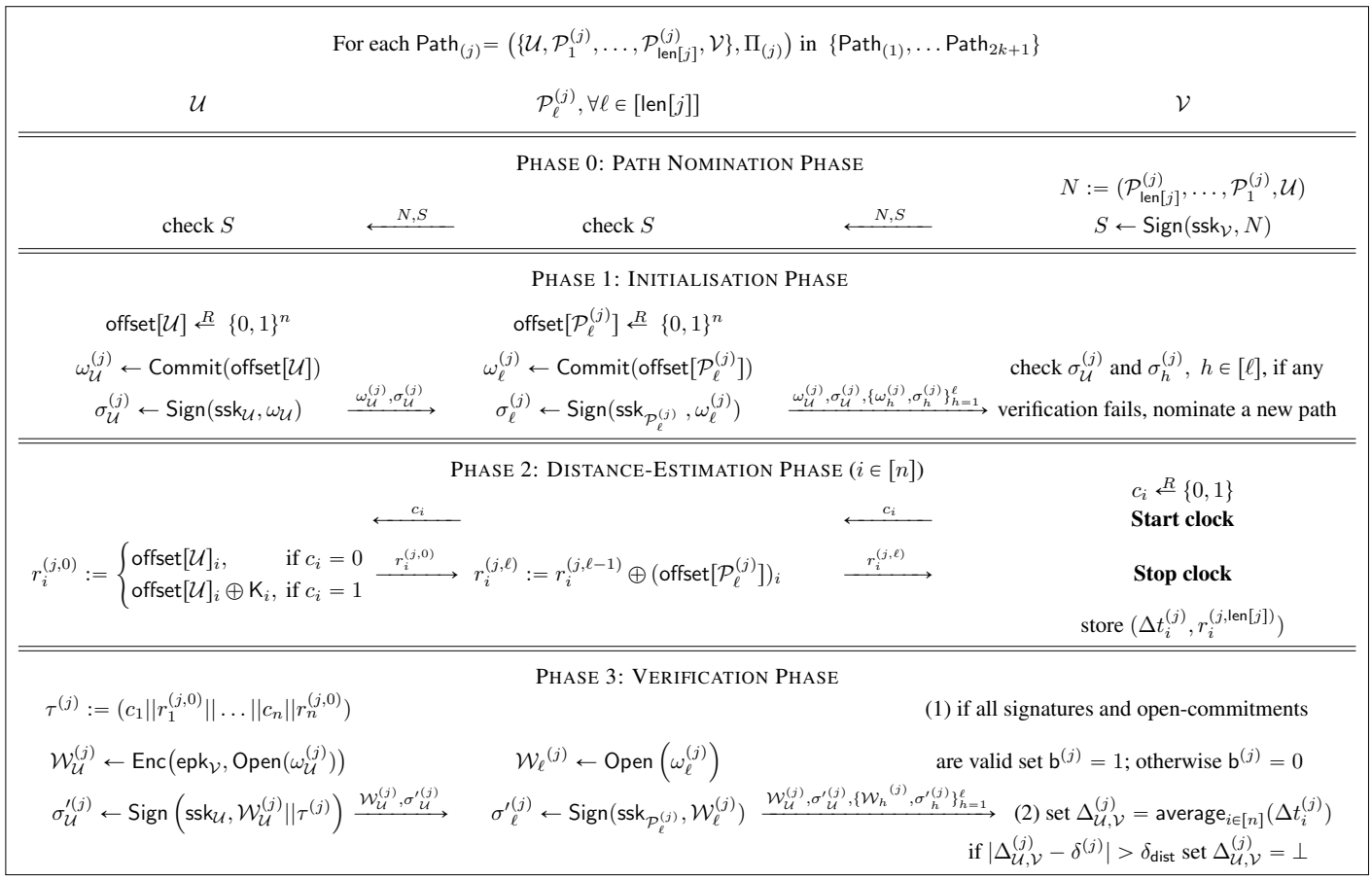


Figure 3: The proposed multi-hop distance estimation (MHDE) protocol.

own offset (generated in the initialisation phase), signs it, and forwards to the next node its opening, the signature and the data received from the previous node.

The verifier's goal is to (1) authenticate the prover $\mathcal{U}$ by checking the correctness of each information exchanged in the protocol, and (2) compute an estimation on the distance between $\mathcal{U}$ and $\mathcal{V}$ using the round-trip values obtained in the time-critical phase. In more details, the verifier goes through the following check list: (i) the signatures on the proxies' opening of the commitments are correct (ii) the prover's signature is correct for the given ciphertext and the the transcript observed by $\mathcal{V}$; (iii) the commitments verify (with respect to binding) given the signed opening information (the verifier can decrypt the opening for $\mathcal{U}$); (iv) the time-critical responses received by the verifier equal the XOR of all the proxies' (including the prover's) offsets and of the corresponding bit of the veriferi-prover shared key K. If all conditions are satisfied, $\mathcal{V}$ sets $\mathsf{b}^{(j)} = 1$, otherwise $\mathsf{b}^{(j)} = 0$. In order to obtain the distance-estimation on Path_j, the verifier computes the average of the measured round-trip times along the path (using the values produced in MHDistEst), and sets the output $\Delta_{\mathcal{U},\mathcal{V}}^{(j)}$ to that average value. This value is checked for consistency with respect to the number of nodes in the path, in the following way. Let δ_ℓ be the communication range of proxy $\mathcal{P}_\ell^{(j)}$, we define the expected distance on Path_j to be $\delta^{(j)} = \sum_{\ell=0}^{\mathsf{len}[j]} \min\{\delta_\ell, \delta_{\ell+1}\}$, where δ_0 and $\delta_{\mathsf{len}[(j)]+1}$ are respectively the communication ranges of $\mathcal{V}$ and $\mathcal{U}$. If $\Delta_{\mathcal{U},\mathcal{V}}^{(j)}$ exceeds the expected distance $\delta^{(j)}$ by more than the accuracy value δ_{dist}, $\Delta_{\mathcal{U},\mathcal{V}}^{(j)}$ is set to $\perp$, otherwise the estimated distance for $\mathsf{Path}_{(j)}$ is set to $\Delta_{\mathcal{U},\mathcal{V}}^{(j)}$.

MHVerify: The MHDE protocol ends with the MHVerify algorithm run by the verifier on the $(\mathsf{b}^{(j)}, \Delta_{\mathcal{U},\mathcal{V}}^{(j)})$ output on each path. If all authentication bits are $1 = \mathsf{b}^{(j)}$, then the verifier sets the global authentication bit $\mathsf{b} = 1$ (else $\mathsf{b} = 0$). The second output of MHVerify is $\Delta_{\mathcal{U},\mathcal{V}}$, the estimated distance between $\mathcal{U}$ and $\mathcal{V}$. From all the paths over which the distance-estimation is done, the verifier will only choose those for which the local estimate $\Delta_{\mathcal{U},\mathcal{V}}^{(j)} \neq \perp$. In our proposal, we choose $\Delta_{\mathcal{U},\mathcal{V}} = \min_{j \in [2k+1]}\{\Delta_{\mathcal{U},\mathcal{V}}^{(j)} : \Delta_{\mathcal{U},\mathcal{V}}^{(j)} \neq \perp\}$. We discuss other functions to extrapolate the final distance estimate from the paths in the next section.

6.3 Accuracy of the distance estimation

To achieve good accuracy levels in the distance estimation, our protocol requires only light-weight operations in the time-critical phase. Indeed each node has to perform (at most) a *read-and-xor*, which reduces any delay in the generation of the

responses. Thus, the two main sources of discrepancies between the actual distance $\mathsf{dist}(\mathcal{U},\mathcal{V})$ and the estimated one $\Delta_{\mathcal{U},\mathcal{V}}$ are: the *shape* of the chosen paths, and the method used to extrapolate $\Delta_{\mathcal{U},\mathcal{V}}$ from the measurements on each path ($\Delta^{(j)}_{\mathcal{U},\mathcal{V}}$). We mitigate the first cause of error by choosing *short* paths, i.e., paths with the least number of nodes. In this way we ensure that the the path is *almost* a straight line between $\mathcal{U}$ and $\mathcal{V}$. Regarding how to compute the final distance estimation, in this work we consider three different approaches: $\Delta_{\mathcal{U},\mathcal{V}}$ is computed as (1) the median among the valid $\Delta^{(j)}_{\mathcal{U},\mathcal{V}}$; (2) the average of these values; and (3) the minimum of these values (this method provides the least error in measurement). PFrom a security point of view, if $\mathcal{V}$ selects the *shortest* measurement, then this value might represent an *under*-estimation of the true distance if adversaries are able to communicate *faster* than honest nodes. Although in many RF scenarios, devices *can* (physcially) communicate at the speed of light, sometimes the standards by which they operate regulate the transmission speeds and frequencies at lower values. Choosing the shortest path does, always, prevent distance-enlarging attacks. Selecting the median value makes the DEE larger, but it also guarantees that the estimated distance comes either from an honest path (a path with only honest users), or that it lies in-between honest-path measurements[4]. A similar reasoning applies to distance-shortening attacks, when the *median* measurement is used. The average value is the easiest to be influenced by malicious nodes. However, our countermeasure of seeking consistency with respect to the number of nodes, significantly bounds the distance estimation error in any case.

7 Security Analysis

In our MHDE protocol we employ a signature scheme $\mathsf{SScheme} = (\mathsf{SKGen}, \mathsf{Sign}, \mathsf{Vf})$ that is existentially unforgeable under chosen-message attacks (EUF-CMA); a public-key encryption scheme $\mathsf{EScheme} = (\mathsf{EKGen}, \mathsf{Enc}, \mathsf{Dec})$ that is indistinguishable under adaptive chosen-ciphertext attacks (IND-CCA). We furthermore require a (computationally) binding and hiding commitment scheme $\mathsf{CScheme} = (\mathsf{Commit}, \mathsf{Open})$.

Under such assumptions, we formulate the following security statement:

Theorem 1. *The protocol* MHDE *described in Section 6 has the following security properties in the presence of an adversary running q executions:*

- *It is $(q, \epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$-resistant to **MiM-attacks**, with:*
  - $\epsilon_{\mathsf{auth}} := q(\mathsf{Adv}^{\mathsf{Unf}}(\mathcal{A}') + \mathsf{Adv}^{\mathsf{IND-CCA}}(\mathcal{A}''))$*, in which $\mathsf{Adv}^{\mathsf{Unf}}(\mathcal{A}')$ denotes the advantage of an adversary $\mathcal{A}'$ we construct against the EUF-CMA of the signature scheme, and $\mathsf{Adv}^{\mathsf{IND-CCA}}(\mathcal{A}'')$ is the corresponding advantage of breaking IND-CCA.*
  - $\delta_{\mathsf{dist}} := (\mathsf{MinLength} + \nu)t_\delta \mathsf{dist}(\mathcal{U},\mathcal{V}) + \mathsf{f}(\mathcal{N})$*, where $\mathsf{dist}(\mathcal{U},\mathcal{V})$ is the real distance between $\mathcal{U}$ and $\mathcal{V}$, and f is an interpolation-error function, which is asymptotically 0 if the minimal path length $\mathsf{MinLength}$ is large*[5]*, while t_δ is the time required to send a message (at broadcast speed) across the one-hop distance δ.*
  - $\epsilon_{\mathsf{dist}} := q(2^{-n} + \mathsf{Adv}^{\mathsf{C.Bind}}(\mathcal{A}') + \mathsf{Adv}^{\mathsf{C.Hide}}(\mathcal{A}'') + \mathsf{Adv}^{\mathsf{Unf}}(\mathcal{A}'''))$*, where $\mathcal{A}'$ is an adversary against the binding of the commitment scheme, $\mathcal{A}''$ breaks the commitment's hiding property, and $\mathcal{A}'''$ is a forger against the signature scheme.*

  If the cheat mode presented in Appendix .4 is also used, ϵ_{auth} and ϵ_{dist} are both increased by: $2^{(-2-\log_2 3)n}q$.
- *It is $(q, \epsilon_{\mathsf{dist}})$-resistant to **distance-shortening**, for $\epsilon_{\mathsf{dist}} = q((\frac{3}{4})^n + \mathsf{Adv}^{\mathsf{C.Bind}}(\mathcal{A}'))$, where $\mathcal{A}'$ is an adversary that breaks the binding property of the commitment scheme.*

Due to space restrictions, the proof of Theorem 1 has been moved to Appendix .5.

7.1 Mitigating the Key-Recovery Attack of [8, 10]

Distance-bounding protocols in which the prover's secret key is XOR-ed with the offset to respond to the verifier's challenges are vulnerable to Kim et al.'s key recovery attack [10]. In our MHDE protocol we mitigate this attack by combining a commitment scheme (on the offset) and a signature scheme (on the transcript). The commitment hides the prover's offset values from $\mathcal{V}$ and from any adversary that does not collude with $\mathcal{U}$. In addition, the signature in the verification phase prevents an attacker from running the key-disclosure attack in [10], as it would require $\mathcal{A}$ to sign a *modified* transcript, which no longer matches the (initially) committed values.

[4] In particular, our choice of using $2k+1$ paths if there are at most k adversaries, and of pruning inconsistent values with respect to the number of hops, guarantee that even if the k corrupted nodes cause k too-large estimates, the median measurement is still below those estimates.

[5] As the network is dense, a path of minimum length is almost a straight line made of the proxies separating $\mathcal{U}$ and $\mathcal{V}$. Higher $\mathsf{MinLength}$ values imply more precise measurements.

7.2 Collusion-Resistance

The protocol presented in Figure 3 does not resist collusion-attacks. As a way to deter $\mathcal{U}^*$ from aiding the colluding proxies, a countermeasure is to force the prover to leak a long-term secret by aiding the proxies. We offer collusion-resistance as an optional property, by introducing a deterring backdoor, similar to the one in [9]. Due to page restrictions, we present this countermeasure in Appendix .4. Although collusion-resistance is a desirable property, it also weakens the overall security of the protocol, since the countermeasure used by the proxies to authenticate on their own can also be exploited by MiM adversaries.

8 Experimental Analysis

An important aspect in the concept of multi-hop distance-estimation is the accuracy of the distance estimate across multiple paths and hops. We experimentally investigate the reliability of the distance-estimation error (DEE) by implementing our protocol in Java[6] for a given (and sufficiently-dense) network topology. We consider nodes in a three-dimensional space, i.e., having three coordinates, and we use Euclidean distances. We employed RSA encryption, DSA signatures, and Pedersen commitments, using $n = 256$ time-critical rounds for each path. These choices are due to simplicity solely, as we want to provide a proof-of-concept simulation to assess the robustness and reliability of our experiments. In particular, the employed RSA encryption is not proved to be IND-CCA and no thorough security proof exists for DSA (although this is widely accepted as a secure scheme). Figure 4 shows the evaluation of the DEE when only honest nodes are involved in the MHDE protocol. Each experiment described below was run 100 times, and the displayed plots are the average obtained by all the 100 independent runs of the program. We evaluate the accuracy of the distance estimated by our MHDE protocol in two types of networks: (i) networks of *homogeneous* devices (in which all nodes have the same communication range $\delta_\ell = \delta = 10$ meters) and (ii) networks of *heterogeneous* devices (the nodes' signal-power-radius is chosen as either 10m, 20m, or 30m). We also perform our experiments for three methods of estimating the prover/verifier distance: (1) the minimum value, (2) the median value, and (3) the average among the paths chosen by $\mathcal{V}$ for the distance estimation. In the sequel, we refer to these approaches as *metrics*. To better simulate real-world scenarios, we exclude the optimal paths from the output of PathGen, i.e., any straight path between $\mathcal{U}$ and $\mathcal{V}$ is discarded by PathGen.

Figure 4(a) depicts the impact on the DEE of the *number* of in-between proxies chosen per path. The fixed distance $\mathsf{dist}(\mathcal{U}, \mathcal{V})$ of 50 meters enforces at least 5 proxies in homogeneous networks (since we discard any straight-line path and the broadcast range is 10m). For the heterogeneous case, the number of proxies per path varies, as the broadcast range is chosen randomly per node. In both cases and for all three metrics considered, the DEE increases with the number of proxies; however, the DEE is always smaller in homogeneous networks. We fix the number of chosen paths to 50, but gradually increase the number of intermediate proxies (between $\mathcal{U}$ and $\mathcal{V}$). The metric of the minimum leads to the best accuracy (lowest DEE) for both types of networks; the DEE is also smaller when the number of proxies along the path is 5 to 9.

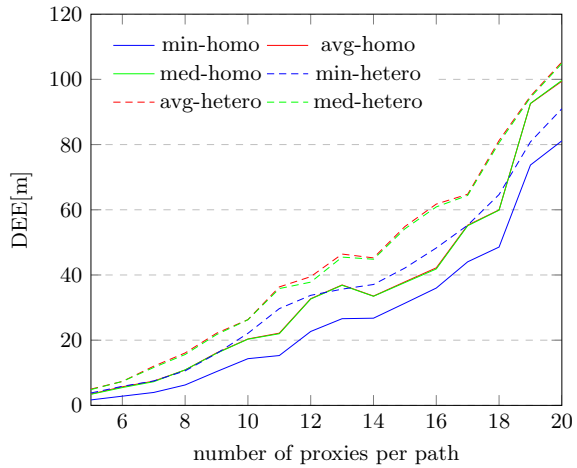
We also investigate the relationship between DEE and the number of chosen *paths*, see Figure 4(b). We choose only paths of length 5 to 9 proxies (as they were optimal in the previous experiment); once more, the DEE produced by the metric of the minimum is the lowest in both network types. Its accuracy in fact increases with the number of chosen paths, whereas no such improvement is registered for the other two metrics. The fact that the DEE appears quite stable with respect to the number of chosen paths for the MHDE (Figure 4(b)) is actually motivated by a very simple observation. Increasing the number of paths does not necessarily provide higher or lower accuracy, since in any case we will be adding both *good* paths (almost straight lines between $\mathcal{U}$ and $\mathcal{V}$) *long* paths. The *long* paths are discarded when the metric of the minimum is employed, and have very little weight in the average of median metrics when the total number of paths increases.

In our third experiment (Figure 4(c)), we analyse the influence of network density on the distance estimation error. Network density is counted in the total number of nodes contained in all paths, for chosen paths of 5 to 9 proxies. As the network density increases, more paths are output by PathGen, yielding a decrease in the DEE. We observe that the DEE is stable with respect to the number of nodes present in the network. This is due to the fact that having more nodes (also lying in-between $\mathcal{U}$ and $\mathcal{V}$) does not affect how our MHDE protocol picks paths. In more details, if $\mathsf{dist}(\mathcal{U}, \mathcal{V})$ = 50 meters, $\mathcal{V}$ will choose (or prefer) paths that have between 5 and 9 nodes; if there are more nodes in the network, this essentially means that there are more potential paths (and especially more straight ones).

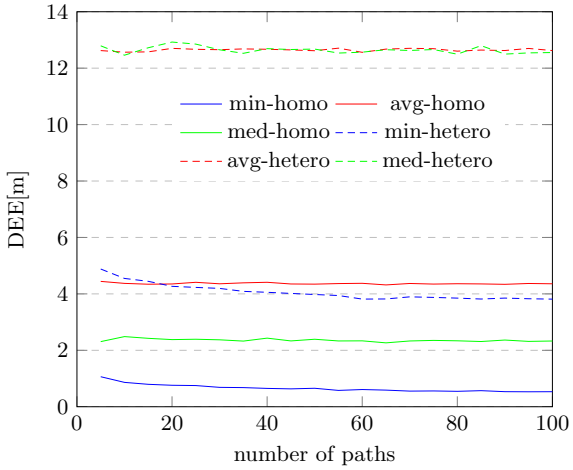
We investigate how the DEE varies with the actual distance between $\mathcal{U}$ and $\mathcal{V}$, see Figure 4(d), for a homogeneous network in which $\mathsf{dist}(\mathcal{U}, \mathcal{V})$ increases in steps of 20 meters from 20 to 200 meters. For all the three metrics considered, the normalised ratio between DEE and the actual distance $\mathsf{dist}(\mathcal{U}, \mathcal{V})$ decreases with the increasing length. The metric of the minimum gives the least error percentage and has the slowest reduction rate.

Our MHDE protocol accommodates a limited mobility of the nodes, as long as the existing paths are not broken. For our last experiment we allow nodes to move (with this restriction) every 15 communication rounds (out of the total 256 ones). As shown in Figure 4(e), the protocol is robust with respect to this mobility, though the DEE increases as the distance between the nodes increases.

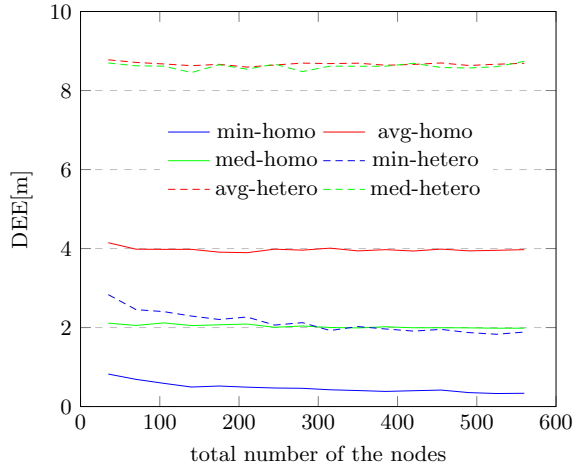
[6] The code used for the simulations is available at `https://drive.google.com/open?id=0B8OW_hrAk_u6ZnRydWczRzJyMEE`.



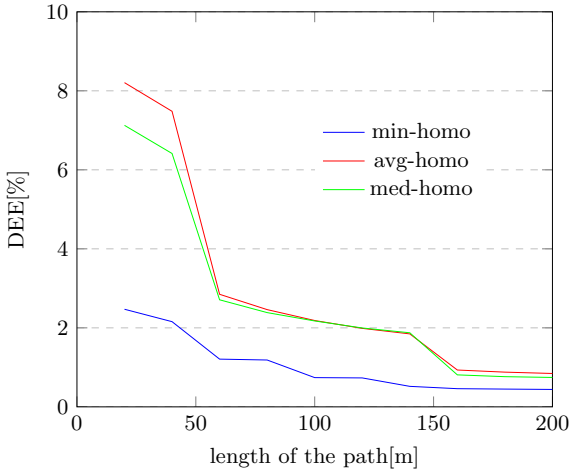
(a) DEE vs. number of proxies chosen per path.



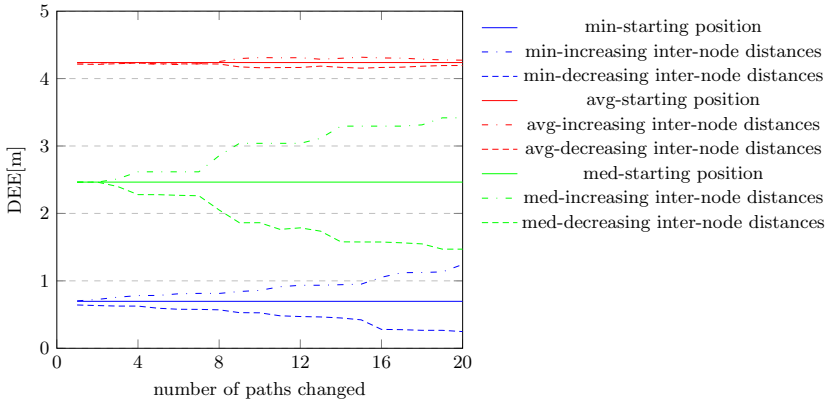
(b) DEE vs. number of total paths chosen.



(c) DEE vs. network density.



(d) DEE percentage vs. length of the path.



(e) DEE vs. node mobility.

Figure 4: Variation of DEE against 4(a) number of proxies chosen per path, 4(b) number of total paths chosen, 4(c) total number of nodes in the network, and 4(e) limited mobility of nodes, when $\mathsf{dist}(\mathcal{U}, \mathcal{V})$ is fixed to 50 meters. Accuracy ratio between DEE and $\mathsf{dist}(\mathcal{U}, \mathcal{V})$ for different values of $\mathsf{dist}(\mathcal{U}, \mathcal{V})$ 4(d).

In our last experiment we analyse how the MHDE protocol performs in networks where malicious proxies have larger signal-power radius than honest users (Figure 5). In more details, we consider homogeneous networks which contain mobile devices each with 10 meters signal-power-radius (δ), and let (an increasing number of) malicious users have a communication range of 30 meters, larger than the claimed δ. Multiple such malicious nodes registered into the network, can collaborate to make $\mathcal{V}$ sub-estimate the value $\mathsf{dist}(\mathcal{U}, \mathcal{V})$. In a successful attack, the distance measured by $\mathcal{V}$ is smaller than the actual distance between $\mathcal{U}$ and $\mathcal{V}$. Figure 5 depicts our simulations in this scenario. The type of attack we mount here is a MiM-Attack. In order to launch such an attack, $\mathcal{A}$ needs to have at least 2 malicious proxies in one (or more)

path(s) chosen by $\mathcal{V}$. Apart from communicating with the respective neighbour nodes in the path, a malicious proxy $\mathcal{P}^*$ can communicate with other malicious proxies in the same path which lie within a 30 meters radius. When this happens, the malicious proxies can send a challenge to $\mathcal{U}$ via the alternative path (formed by the malicious proxies) before the challenge is received via the path chosen initially by $\mathcal{V}$. It should also be noted that, we place a malicious proxy as the next immediate neighbour of $\mathcal{U}$ in a considered path so that it can coordinate the messages appropriately and avoid the honest proxies or $\mathcal{V}$ detecting the malicious behaviour. For this experiment, we consider only the metric minimum, since it is the metric which gives the least DEE. It is notable that none of the mounted MiM-attacks was successful in making the verifier sub-estimate the actual distance between $\mathcal{V}$ and $\mathcal{U}$, rather, the attacks yield to a reduction in the DEE (i.e., higher accuracy).

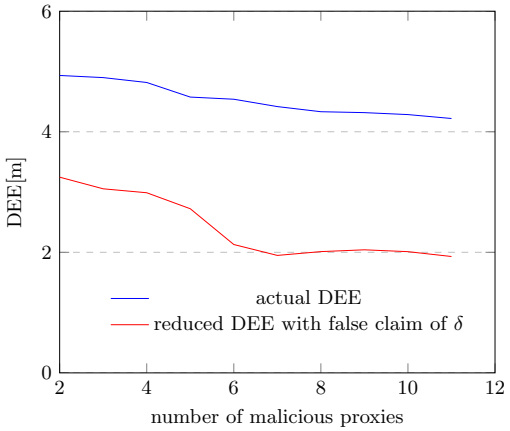


Figure 5: Variation of DEE against false claim of signal-power-radius (δ).

9 Discussion & Further Work

In this paper, we introduced the new concept *multi-hop distance estimation* (MHDE) and proposed a novel MHDE protocol to achieve multi-hop distance-estimation in ad-hoc communication networks made of pervasive devices. Our protocol extends the notion of distance-bounding to a new level as it introduces the concept of distance estimation across several nodes and different paths. The proposed MHDE protocol is a proof-of-concept for multi-hop distance-bounding; it guarantees high levels of security against malicious provers and even under collusion. The accuracy of the distance estimation depends on the choice of the metric, and our experiments show that it is possible to keep it within a small fraction of the actual distance between the two main entities of the protocol (the prover and the verifier). We tested the accuracy of the distance estimation of the proposed MHDE protocol from several points of view, and established it is mostly influenced by the density of the network. In particular, only a *high* density network gives the verifier the opportunity to choose a *nearly straight* path (from it to the prover), over which to estimate the distance. In addition, we analysed how the distance estimation error variates when the verifier chooses different numbers of paths, paths with more or fewer proxies and when the proxies have different signal-power-radius. We demonstrated that our MHDE protocol is robust with respect to *limited* movement of the proxies, i.e., as long as it does not affect the network topology on the chosen paths. For use cases where the network is densely populated, the protocol is even robust with respect to some modifications of the network structure itself. This covers the situations in which proxies move at a limited speed.

It would be worth to analyse the performance of our MHDE protocol under the presence of *noisy* communication, and we keep this analysis for a stand-alone study. Another interesting direction for future work is to investigate how far one can use a public-key approach for MHDE so as to improve the privacy of our construction. Additionally, one could consider that the network topology is generated from an Erdös Renyi model [7] and characterise the gap between the shortest path and the expected shortest path free of malicious nodes.

References

[1] G. Avoine, M. A. Bingol, S. Karda, C. Lauradoux, and B. Martin. A formal framework for analyzing RFID distance bounding protocols. In *Journal of Computer Security - Special Issue on RFID System Security, 2010*, 2010.

[2] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Towards secure distance bounding. In *Proceedings of FSE'13*, volume 8424 of *LNCS*, pages 55–67. Springer-Verlag, 2013.

[3] S. Brands and D. Chaum. Distance-bounding protocols. In *Proc. of EUROCRYPT'93*, volume 765 of *LNCS*, pages 344–359. Springer-Verlag, 1993.

[4] S. Capkun and J.-P. Hubaux. Secure positioning in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):221–232, Feb 2006.

[5] N. Chandran, V. Goyal, R. Moriarty, and R. Ostrovsky. Position based cryptography. In *Proc. of Advances in Cryptology – CRYPTO 2009*, volume 5677 of *LNCS*, pages 391–407. Springer-Verlag, 2009.

[6] U. Dürholz, M. Fischlin, M. Kasper, and C. Onete. A formal approach to distance bounding RFID protocols. In *Proc. of ISC'11*, volume 7001 of *LNCS*, pages 47–62. Springer-Verlag, 2011.

[7] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6, 1959.

[8] M. Fischlin and C. Onete. Subtle kinks in distance-bounding: an analysis of prominent protocols. In *Proc. WISec'13*, pages 195–206. ACM Press, 2013.

[9] M. Fischlin and C. Onete. Terrorism in distance bounding: Modeling terrorist fraud resistance. In *Proc. of ACNS'13*, volume 7954 of *LNCS*, pages 414–431. Springer-Verlag, 2013.

[10] C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira. The swiss-knife rfid distance bounding protocol. In *Information Security and Cryptology–ICISC 2008*, pages 98–115. Springer, 2008.

[11] M. Kuhn. An asymmetric security mechanism for navigation signals. In J. Fridrich, editor, *Information Hiding*, volume 3200 of *Lecture Notes in Computer Science*, pages 239–252. Springer Berlin Heidelberg, 2005.

[12] L. Lazos, R. Poovendran, and S. Čapkun. Rope: Robust position estimation in wireless sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[13] K. B. Rasmussen and S. Capkun. Implications of radio fingerprinting on the security of sensor networks. In *Proceedings of the International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*, 2007.

[14] R. Shokri, M. Poturalski, G. Ravot, P. Papadimitratos, and J.-P. Hubaux. A practical secure neighbor verification protocol for wireless sensor networks. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 193–200, New York, NY, USA, 2009. ACM.

[15] A. Srinivasan and J. Wu. A survey on secure localization in wireless sensor networks. *Encyclopedia of Wireless and Mobile Communications*, 2008.

[16] S. Vaudenay. On modeling terrorist frauds - addressing collusion in distance bounding protocols. volume 8209 of *LNCS*, pages 1–20. Springer-Verlag, 2014.

[17] X. Zheng, R. Safavi-Naini, and H. Ahmadi. Distance lower bounding. In *16th International Conference on Information and Communications Security (ICICS 2014)*, Hong Kong, December 2014.

.1 A Path-Generation Algorithm

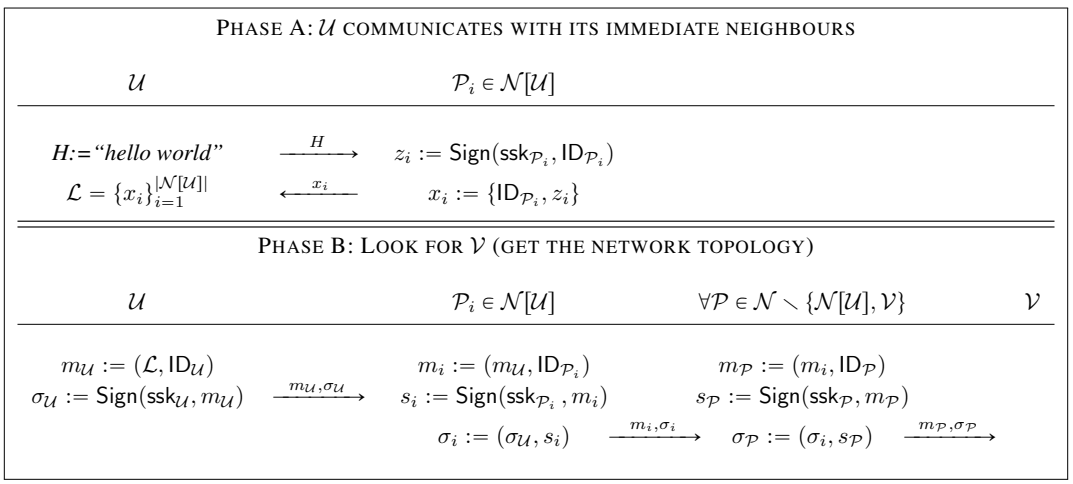


Figure 6: A topology-establishing algorithm between $\mathcal{U}$ and $\mathcal{V}$.

As a preamble of our MHDE protocol, we run a path-generation algorithm, in which a verifier learns something about the topology of the current network, namely by finding paths to the prover such that: (1) each node on a path really *is* on that path; (2) the order of the nodes is genuine. While this step is orthogonal to the main contribution of our work and could be integrated within the routing protocol, we give an example below of how path generation can be achieved (see also Figure 6). Our protocol has two phases. In (PHASE A) the prover contacts its immediate neighbours, while in (PHASE B) messages are transmitted from one part to its one-hop neighbours until they reach $\mathcal{V}$. By the end of the second phase, $\mathcal{V}$ shall have enough information for the network topology to construct paths connecting $\mathcal{U}$ to $\mathcal{V}$.

More concretely, PHASE A is composed of the following steps. A "Hello world" message is sent by $\mathcal{U}$, and its 1-hop neighbours broadcast their ID, with a corresponding signature. The prover stores each ID/signature pair for which the signature verifies in a list $\mathcal{L}$. In PHASE B, the prover $\mathcal{U}$ sends its identity with a signature on it to its one-hop neighbours. These neighbours append their identity to this message, sign the concatenation of the two, and forward it to *their* one-hop neighbours. This is repeated from one proxy to its neighbours, until the messages reach the verifier $\mathcal{V}$. The latter verifies all the signatures; if the verification fails, the corresponding message is dropped. Using the remaining signatures, the verifier $\mathcal{V}$ constructs paths to the prover.

Note: Role of the Signatures In our MHDE protocol, for each of the generated paths, the verifier needs a proof $\Pi^{(j)}$ that (1) each of the nodes indicated in the path actually participated in creating the path; (2) the order of the nodes is the correct one. This proof consists in fact of the sequence of messages received by the verifier and the concatenation of signatures. If any of these signatures do not verify, the proof of the path is rejected (in particular, the verifier will not choose that path to do the distance-estimation step).

.2 Protocol syntax

A *multi-hop distance-estimation* protocol over a network $\mathcal{N}$ is defined by the following algorithms:

$\mathsf{MHSetup}(1^\lambda) \rightarrow \mathsf{ppar}$ The Setup algorithm takes as input a security parameter λ and outputs the global (public) parameters ppar.

$\mathsf{MHUReg}(\mathcal{P}) \rightarrow (\mathsf{spar}[\mathcal{P}], \mathsf{ppar}[\mathcal{P}])$ The User Registration algorithm sets up an input user $\mathcal{P}$ to be a proxy (by default), outputting secret parameters $\mathsf{spar}[\mathcal{P}]$ and public parameters $\mathsf{ppar}[\mathcal{P}]$.

$\mathsf{MHSelect}(\mathcal{P}_i, \mathcal{P}_j) \rightarrow (\mathsf{esk}, \mathsf{epk}, \mathsf{K})$ This algorithm allows for the setup of two proxies as prover and verifier respectively. These users will then share a secret key K generated at random, and the verifier will be associated by a tuple of encryption keys $(\mathsf{esk}, \mathsf{epk})$. This step is run every time either the prover or the verifier changes.

$\mathsf{PathGen}(\mathcal{N}, \mathcal{U}, \mathcal{V}) \rightarrow \{\mathsf{Path}_\ell\}_{\ell \in L}$ The path-generation algorithm $\mathsf{PathGen}$ takes as input the current network $\mathcal{N}$ and the identifiers of the prover and verifier respectively. It outputs a list of paths from the prover to the verifier.

$\mathsf{MHProve}$**,** $\mathsf{MHProxy}$**,** $\mathsf{MHVerifier}$ These are, respectively, the algorithms used by the prover, the proxies, and the verifier in the main part of the protocol. These algorithms interact with each other, and each party uses its secret parameters and the public network parameters as input.

$\mathsf{MHVerify}$ The verification algorithm takes as input the protocol transcript, $\mathsf{spar}[\mathcal{V}]$, the public parameters of the network $\mathsf{ppar}[\mathcal{N}]$ and the output of the $\mathsf{PathGen}$ algorithm. It has two outputs: an authentication bit $\mathsf{b} \in \{0, 1\}$ (b=1 if the prover is authenticated), and an estimation $\Delta_{\mathcal{U},\mathcal{V}}$ of the physical distance $\mathsf{dist}(\mathcal{U}, \mathcal{V})$.

Our definition of MHDE allows the protocol to return different outputs when run between a prover and a verifier placed at the same distance. The output of the distance-estimation depends on the position of the proxies in $\mathcal{N}$; the transcripts of the protocol may also differ in the two executions.

.3 A Detailed Description of the MHDE Protocol

This appendix provides a detailed description of the protocol proposed in this paper. For each path Path_j (selected by $\mathcal{V}$ from the output of $\mathsf{PathGen}$), $j \in [2k+1]$, the MHDE protocol has the following four phases:

(i) PATH NOMINATION: the verifier broadcasts and signs one $\mathsf{Path}_{(j)} \in \mathcal{D}$. The path is propagated in the inverse order of hops from $\mathcal{V}$ back to $\mathcal{U}$.

(ii) INITIALISATION: this is a non-time-critical phase, in which each intermediate node (proxy) $\mathcal{P}_\ell^{(j)} \in \mathsf{Path}_{(j)}$selects uniformly at random an n-bit string called $\mathsf{offset}[\mathcal{P}_\ell^{(j)}] \in \{0,1\}^n$. The offset is then committed in $\omega_\ell^{(j)} \leftarrow \mathsf{Commit}(\mathsf{offset}[\mathcal{P}_\ell^{(j)}])$, and $\omega_\ell^{(j)}$ is signed, $\sigma_\ell^{(j)} \leftarrow \mathsf{Sign}(\mathsf{ssk}_\mathcal{P}, \omega_\ell^{(j)})$. Eventually, the signature and the commitment are forwarded to the *next node*[7] in the path. Upon receiving a message, each node relays it to the next node, until all pairs $\{(\omega_\ell^{(j)}, \sigma_\ell^{(j)})\}_{\ell=1}^{\mathsf{len}[j]}$ reach

[7] For $\ell \in [\mathsf{len}[j]-1]$, $\mathcal{P}_\ell^{(j)}$'s next node is $\mathcal{P}_{l+1}^{(j)}$. At the extremes of the path, $\mathcal{U}$'s next node is $\mathcal{P}_1^{(j)}$, while $\mathcal{P}_{\mathsf{len}[j]}^{(j)}$'s next node is $\mathcal{V}$.

$\mathcal{V}$. (iii) DISTANCE ESTIMATION: In this time-critical phase, $\mathcal{V}$ picks a 1-bit random challenge $c_i \in \{0,1\}$, starts the clock, and sends c_i to the first node $\mathcal{P}^{(j)}_{\mathsf{len}[j]}$ in the selected path $\mathsf{Path}_{(j)}$. Each node $\mathcal{P}^{(j)}_{\ell}$ forwards c_i to the *next* node $\mathcal{P}^{(j)}_{\ell-1}$ in $\mathsf{Path}_{(j)}$ until c_i reaches $\mathcal{U}$. The prover, $\mathcal{U}$, computes the response

$$r_i^{(j,0)} := \begin{cases} \mathsf{offset}[\mathcal{U}]_i, & \text{if } c_i = 0 \\ \mathsf{offset}[\mathcal{U}]_i \oplus \mathsf{K}_i, & \text{if } c_i = 1 \end{cases}$$

using the secret key K (shared between the prover $\mathcal{U}$ and the verifier $\mathcal{V}$). Finally, $\mathcal{U}$ sends $r_i^{(j,0)}$ to the next node in the path $\mathsf{Path}_{(j)}$. For $\ell \in [\mathsf{len}[j]]$, upon receiving a bit $r_i^{(j,\ell-1)}$, node $\mathcal{P}^{(j)}_{\ell}$ computes

$$r_i^{(j,\ell)} := r_i^{(j,\ell-1)} \oplus (\mathsf{offset}[\mathcal{P}^{(j)}_{\ell}])_i$$

and forwards the resulting bit to the next node in the path. As soon as $\mathcal{V}$ receives $r_i^{(j,\mathsf{len}[j])}$ from $\mathcal{P}^{(j)}_{\mathsf{len}[j]}$, the verifier stops the clock and stores the value $r_i^{(j,\mathsf{len}[j])}$ together with $\Delta t_i^{(j)}$, which corresponds to the time elapsed between sending the challenge c_i and receiving the response $r_i^{(j,\mathsf{len}[j])}$ in path $\mathsf{Path}_{(j)}$.

(iv) VERIFICATION: The final phase of the MHDE protocol has two main purposes: *authenticate* the prover and *estimate the distance* between $\mathcal{U}$ and $\mathcal{V}$. The former goal is achieved by checking the correctness of the responses collected during the distance-estimation phase. More precisely, each proxy $\mathcal{P}_\ell \in \mathsf{Path}_{(j)}$. In order to prevent the leakage of the shared secret K, the prover *encrypts* the opening of its commitment with the verifier's public key, obtaining $\mathcal{W}^{(j)}_{\mathcal{U}} := \mathsf{Enc}(\mathsf{epk}_{\mathcal{V}}, \mathsf{Open}(\omega^{(j)}_{\mathcal{U}}))$. Then, $\mathcal{U}$ computes the signature $\sigma'^{(j)}_{\mathcal{U}} := \mathsf{Sign}\left(\mathsf{ssk}_{\mathcal{U}}, \mathcal{W}^{(j)}_{\mathcal{U}} || \tau^{(j)}\right)$ on the concatenation of the encryption of the open commitment and the transcript $\tau^{(j)}$. After collecting the messages $\mathcal{W}^{(j)}_{\ell}, \sigma'^{(j)}_{\ell}$ from all the $\mathcal{P}^{(j)}_{\ell} \in \mathsf{Path}_{(j)}$the verifier checks if all signatures are valid, in which case $\mathcal{V}$ decrypts $\mathcal{W}^{(j)}_{\mathcal{U}}$ to obtain $\mathsf{offset}[\mathcal{U}]$. Subsequently, $\mathcal{V}$ controls if the initial commitments correspond to the given openings. Finally, for each time-critical round on $\mathsf{Path}_{(j)}$, the verifier checks that the responses $r_i^{(j,\mathsf{len}[j])}$ are consistent with the challenges and the offset values. If any of the previous checks fails $\mathcal{V}$ returns authentication failure, i.e., the authentication bit is set to $b^{(j)} = 0$ and aborts. If $\mathcal{U}$ is authenticated, $\mathcal{V}$ proceeds to compute the distance $\Delta_{\mathcal{U},\mathcal{V}}$ according to the round-trip times obtained during the time critical phase. That is, the verifier sets $\Delta^{(j)}$ to be the average of the round-trip times $\Delta t_i^{(j)}$ recorded during Phase (iii).

.4 Threat Model

This appendix contains a detailed description of the adversarial model and of the security notions proposed in the main paper.

The adversary $\mathcal{A}$ is given access to the following oracles:

Network initialisation $\mathsf{InitN}(\cdot,\cdot)$**:** this oracle takes as input an adversarially-chosen topology τ, a set of user identities including the prover $\mathcal{U}$, the verifier $\mathcal{V}$ and the set $\mathbf{P}^*$ of malicious proxies. InitN aborts if the network $\mathcal{N}$ generated by τ does not fulfil the conditions of Section 3. Otherwise, the oracle sets up a network $\mathcal{N}$ according to τ by running MHSetup, MHUReg and MHSelect($\mathcal{V}$,$\mathcal{U}$). Finally, the oracle returns all the public user parameters in $\mathsf{ppar}[\mathcal{N}]$ to $\mathcal{A}$, as well as all the secret parameters of the users in $\mathbf{P}^*$ ($\mathcal{U}$ can belong to $\mathbf{P}^*$). All the other oracles below will return an error symbol $\perp$ if this oracle has never been run. For administration purposes, the security games keep track of the set of malicious users, the current topology, and the current prover and verifier.

Session generation $\mathsf{NewSession}()$**:** the adversary can prompt this oracle (with no input) in order to start a new session of the MHDE protocol. The implicit input is the current network topology $\mathcal{N}$ and the current identities of the prover $\mathcal{U}$ and the verifier $\mathcal{V}$; then NewSession runs the PathGen algorithm internally and uses it in MHVerifier to select the paths to be used in the protocol run. The NewSession oracle returns to the adversary the set of paths chosen by $\mathcal{V}$ together with the π_s of the new session.

Sending $\mathsf{Send}(\cdot,\cdot,\cdot)$**:** this oracle can be invoked in a currently running session only, it takes as input a message m and two proxies identities $\mathcal{P}$ and $\mathcal{P}'$. Send will deliver m *to* a user $\mathcal{P}$ as a message *from* $\mathcal{P}'$. This oracle allows malicious proxies to unicast to any other user, impersonating any proxy in $\mathcal{N}$. A special dedicated message 'prompt' can be sent to $\mathcal{U}$ to make the prover initialise the protocol.

Reset $\mathsf{Reset}(\cdot,\cdot,\cdot)$**:** this oracle allows $\mathcal{A}$ to change the topology of the initial network $\mathcal{N}$ to another topology also fulfilling the conditions outlined in Section 3. The second and third parameters are users $\mathcal{P}$ and $\mathcal{P}'$, which are identifiers of a new prover and a new verifier. If the prover and verifier are different from the current ones, the MHSelect algorithm is run to output new prover and verifier parameters. Note that in particular $\mathcal{A}$ can relocate (but not change) the malicious nodes, and also move honest nodes. When this oracle is used, all current protocol sessions are automatically aborted (we do not allow the topology to change during a protocol run). Furthermore, the adversary can choose malicious nodes to be the prover and the verifier.

Result $\mathsf{Result}(\cdot)$**:** given a valid session of the MHDE protocol π_s, the result oracle returns to $\mathcal{A}$ the verifier's outputs, i.e. the authentication bit b together with the distance estimation $\Delta_{\mathcal{U},\mathcal{V}}$ between $\mathcal{U}$ and $\mathcal{V}$.

InitN is the only oracle that can be called once per session, all other oracles can be queried multiple times. Unless differently specified, in all the security games, we assume that the network $\mathcal{N}$ is initialised with an honest run of MHSetup, MHUReg and MHSelect.

Definition ($(q, \epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$-MiM -Resistance)**.** *A* MHDE *protocol is* $(q, \epsilon_{\mathsf{auth}}, \delta_{\mathsf{dist}}, \epsilon_{\mathsf{dist}})$*-MiM-Resistant if, for all adversaries* $\mathcal{A}$ *making at most* q *queries to the* NewSession *oracle, it holds that if it holds that:*
$\mathsf{Prob}\left[\exists j \in \{1, ..., q\} \text{ s.t. } \mathsf{Result}(\pi_j) = (\mathsf{b} = 1, \Delta_{\mathcal{U},\mathcal{V}})\right] \geqslant \epsilon_{\mathsf{auth}}$ *for a session* π_j *in which both the prover and the verifier were honest, then both the following are satisfied:*

1. *There exists a* $\mathsf{Send}(m = \mathsf{prompt}, \mathcal{U})$ *query that the adversary made to the prover in session* π_j*;*
2. *It holds that the estimated distance* $\Delta_{\mathcal{U},\mathcal{V}}$ *computed in session* π_j*, is such that:* $\mathsf{Prob}\left[\ |\mathsf{dist}(\mathcal{U}, \mathcal{V}) - \Delta_{\mathcal{U},\mathcal{V}}| \geqslant \delta_{\mathsf{dist}}\right] \geqslant 1 - \epsilon_{\mathsf{dist}}$ *for this session.*

Definition ($(q, \epsilon_{\mathsf{dist}})$-DS-Resistance)**.** *A multi-hop distance-estimation authentication protocol* MHDE *is* $(q, \epsilon_{\mathsf{dist}})$*-Distance-Shortening-resistant if, for all adversaries* $\mathcal{U}^*$ *running at most* q *protocol sessions* π_j*,* $j \in [q]$*, there exists a session for which (with probability higher than* ϵ_{dist}*) the output of* MHVerify *is* $(\mathsf{b} = 1, \Delta_{\mathcal{U}^*,\mathcal{V}})$ *with* $\Delta_{\mathcal{U}^*,\mathcal{V}} < \mathsf{dist}(\mathcal{U}^*, \mathcal{V})$*.*

Definition ($(\epsilon_{\mathsf{auth}})$-failed MP-phase)**.** *We consider a pair of adversaries* $\mathcal{A}$ *and* $\mathcal{A}'$*, playing the two-phase collusion game. In particular,* $\mathcal{A} := \mathbf{P}^*_{\mathrm{MP}}$ *with* $\mathcal{U}^* \in \mathbf{P}^*_{\mathrm{MP}}$*, plays the malicious-prover phase (MP-phase); we allow* $\mathcal{A}$ *to run at most* q *protocol sessions against honest verifiers, which we label* $\pi_1^{\mathcal{A}}, \ldots \pi_q^{\mathcal{A}}$*. After the* q *sessions,* $\mathcal{A}$ *passes its views* $\mathsf{view}_{\mathcal{P}^*}$ *for each* $\mathcal{P}^* \in \mathbf{P}^*_{\mathrm{MP}}$ *to the adversary* $\mathcal{A}' := \mathbf{P}^*_{\mathrm{HP}} \supseteq \mathbf{P}^*_{\mathrm{MP}}$*. The latter subsequently plays the honest-prover phase (HP-phase) (with an equal number* q *of sessions, labelled* $\pi_1^{\mathcal{A}'}, \ldots \pi_q^{\mathcal{A}'}$*), again for honest verifiers.*

The goal of $\mathcal{A}$ *is to authenticate to the verifier* and *to make* $\mathcal{V}$ *sub-estimate the distance* $\mathsf{dist}(\mathcal{U}^*, \mathcal{V})$*. The goal of* $\mathcal{A}'$ *is simply to authenticate to the verifier.*

We say that $\mathcal{A}$ *has* failed *its MP-phase (with respect to* $\mathcal{A}'$*) if either of the following holds:*

- *For all sessions* $\pi_j^{\mathcal{A}}$ *in the MP-phase with* $j \in [q]$ *it holds that either the output of* MHVerify *in session* $\pi_j^{\mathcal{A}}$ *is* $(0, \Delta_{\mathcal{U}^*,\mathcal{V}})$ *with probability* ϵ_{auth}*, or, the output of* MHVerify *in session* $\pi_j^{\mathcal{A}}$ *is* $(1, \Delta_{\mathcal{U}^*,\mathcal{V}})$ *and* $\mathsf{Prob}\left[|\Delta_{\mathcal{U}^*,\mathcal{V}} - \mathsf{dist}(\mathcal{U}^*, \mathcal{V})| > \delta_{\mathsf{dist}}\right] \leqslant \epsilon_{\mathsf{auth}}$.
- *With probability higher than* ϵ_{auth}*, there exists a session* $\pi_j^{\mathcal{A}}$ *in the HP-phase, for which the output of* MHVerify *in session* $\pi_j^{\mathcal{A}}$ *has an authentication bit* $\mathsf{b} = 1$ *and* $\mathcal{U}$ *did not participate to the protocol run of session* $\pi_j^{\mathcal{A}}$*.*

We note that this definition of *failed* MP-phase resembles that of *helpfulness* in the context of terrorist-fraud resistance [9].

Definition ($(q,p,\epsilon_{\mathsf{auth}})$-Collusion-Resistance)**.** *A multi-hop distance-estimation authentication protocol* MHDE *is* $(q, \epsilon_{\mathsf{auth}})$*-Collusion-resistant if, for all sets of dishonest entities* $\{\mathcal{U}^*, \mathbf{P}^*\}$ *playing an MP-collusion attack with* q *sessions and winning with non-negligible probability* p*, there exists an HP phase adversary* $\mathcal{A} \subseteq \mathbf{P}^*$ *with respect to which the proxies* $\{\mathcal{U}^*, \mathbf{P}^*\}$ ϵ_{auth}*-failed the MP-phase.*

Cheat Modes. In order to ensure collusion resistance in MHDE, we have to build a back-door into the protocol. This enables us to control how much secret information (i.e., the secret key K), the malicious prover $\mathcal{U}^*$ discloses to its accomplices (the adversary) in order to ensure that the latter can succeed in impersonating $\mathcal{U}$ in the future. As this setting weakens the security of our final result, we consider it as an optional feature to be added in case collusions are likely to happen.

Our construction uses a mechanism similar to the one proposed by Fischlin and Onete in [9]. In our setting, however, the cheat model requires the prover to leak all its secrets (i.e., $\mathsf{spar}[\mathcal{U}]$ and $\mathsf{ssk}_{\mathcal{U}}$). This approach is similar to the 'all-or-nothing' security. We achieve collusion resistance by modifying the MHDE protocol as follows: if the prover receives a request of the form $1||\mathsf{K}'$, where K' is a *close* guess of K, i.e. the Hamming weight of $\mathsf{K} \oplus \mathsf{K}'$ is *small*, the prover outputs its full secret parameters $\mathsf{spar}[\mathcal{U}]$. This is described in Figure 7, where $\mathcal{A}$ denotes a (potentially malicious) entity.

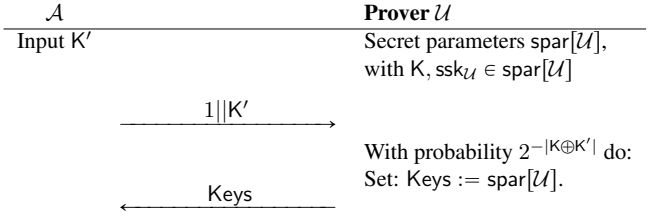


Figure 7: Cheating modes: key disclosure.

In particular, this introduces the possibility of prover impersonation for parties that know (or can guess) a significantly-large part of K.

.5 Proof Sketches

In what follows, we provide the intuition behind the proof of Theorem 1 in Section 7.

Proof.

MiM-Attacks. We briefly outline the proof strategy. Recall that for an MiM attack, the prover and the verifier are honest, but we consider an adversary consisting of at most k dishonest proxies. We make the following observations:

(1) If the prover does not take part in a specific session (no prompting message is sent), then the adversary $\mathcal{A}$ cannot authenticate (thus, making the verifier output an authentication bit $\mathsf{b} = 1$) unless it forges the signature at the end of the transcript. This accounts for the first bound, for ϵ_{auth};

(2) If a prompting request *is* sent, the prover may be authenticated by the verifier; however, even malicious users cannot send messages faster than at the speed of light.

The latter consideration leads us to the following analysis. In order to "shorten" the distance estimation between the prover and the verifier, the adversaries have the following strategies:

(i) Guess the offset values hidden in commitments (this accounts for the adversary $\mathcal{A}''$);

(ii) Cheat on the commitment value (manage to get away with a fake answer) – this accounts for $\mathcal{A}'$;

(iii) Guess the verifier's challenges in advance, accounting for the factor $\frac{1}{2}$ per time-critical round (or, equivalently, guessing the responses);

(iv) Guess at least one challenge/response wrongly and forge the signature at the end (accounting for $\mathcal{A}'''$).

Now we consider the super-estimation of the distance between the two parties. Note that we consider a dense network in which the verifier always chooses the shortest paths (of length $\mathsf{MinLength}$ to $\mathsf{MinLength} + \nu$ for the tolerance rate ν). The triangle inequality enables us to have a spline-like interpolation of the true distance, which is upper-bounded by $(\nu + \mathsf{MinLength})t_\delta \mathsf{dist}(\mathcal{U}, \mathcal{V})$. The interpolation rate is expected to be asymptotic in the topology of the network since the farther the prover is from the verifier, the higher number of possible paths the verifier will have and the likelier $\mathcal{V}$ will choose a path that is "straighter", i.e., closer to the interpolation.

Finally, if cheat modes are used, the adversary also has a probability of guessing (close enough) the value of K and thus retrieve the secret parameters $\mathsf{spar}[\mathcal{U}]$ (making $\mathcal{A}$ able to impersonate the prover). In fact, for any given guess K' of K, the probability that the adversary is given the secret parameters is $2^{-|\mathsf{K} \oplus \mathsf{K}'|}$, where $|\mathsf{K} \oplus \mathsf{K}'|$ denotes the Hamming distance of the two values. Summing over all the possible guesses, we have a final success probability $2^{(-2-\log_2 3)n}$ per session with the prover.

Distance-Shortening. Note that in this case the adversary is (exclusively) the malicious prover. We concretely assumed that K is chosen honestly (independently and uniformly at random), thus, the probability that any one bit of K is 0 is $\frac{1}{2}$. In particular, for any challenge c_i, since the Hamming distance of the two possible responses is precisely equal to the corresponding bit of the key K, the prover has the following alternatives (for each round):

(i) The prover guesses that the challenge will be a particular value c and sends the corresponding response. The guess is correct with probability $\frac{1}{2}$; if the guess is incorrect, the two responses are equal with probability another $\frac{1}{2}$, totalling $\frac{3}{4}$ per round;

(ii) The adversary (malicious prover) can open the commitment to a different value than the committed offset (accounting for the binding advantage). This accounts for the given bound.

Collusion-Resistance. Note that if the adversary (malicious prover colluding with k proxies) wins the malicious prover phase, this implies that the estimated distance must at the end be inferior to the real distance. As discussed for MiM attacks, this implies one of two things:

(i) Either the prover has guessed the challenge (and sent to a proxy closer to the verifier the correct response), or sent the response in advance;

(ii) Or the prover sends both possible responses, enabling the adversary to recover the corresponding bit of the secret key.

This is formally used as follows. We assume the existence of a successful adversary-proxy collusion in the malicious prover phase. Given the views of the respective proxies now the honest-prover adversary $\mathcal{A}'$ runs the malicious adversary in a black box for both challenges and recovers the responses. If the adversary $\mathcal{A}$ (run as a black-box) returns values r_i and r'_i for the two challenges, $\mathcal{A}'$ sets $\mathsf{K}'_i := r_i \oplus r'_i$ (this is a guess of bit K_i). If the adversary $\mathcal{A}$ is unable to respond on at least one of the branches, $\mathcal{A}'$ sets K'_i at random.

As in [9], we show that the probability of $\mathcal{A}'$ to learn the secret parameters $\mathsf{spar}[\mathcal{U}]$ (by using the cheat mode, using the guess K' of K) is equal to $\mathcal{A}$'s probability to succeed, which means $\mathcal{A}'$ authenticates with probability at least p.