UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Tamor Tomson

Toward the Practical Realization of Decentralized Proof-of-Location Systems

Bachelor's Thesis (9 ECTS)

Supervisors:
Amnir Hadachi, PhD
Eduardo Brito, MSc

Tartu 2026

Toward the Practical Realization of Decentralized Proof-of-Location Systems

Abstract:

This thesis addresses the lack of real-world validation for decentralized Proof-of-Location (PoL) systems by designing and experimentally evaluating the first physical prototype of the witnessing-zone model. Developed on low-cost commodity hardware, the system integrates ESP32-DWM3000 ultra-wideband modules for distance bounding, Raspberry Pi 4 nodes for B.A.T.M.A.N.-Adv mesh networking, and a GoQuorum Istanbul BFT blockchain for tamper-evident claim storage. To adapt the protocol to practical hardware constraints, the work introduces a witness-initiated ranging exchange and a cryptographic block-hash binding to defeat replay attacks at the physical layer. Results from a campaign of 340 trials demonstrate a mean position error of 0.18 m and a 100% acceptance rate under honest operation, while successfully detecting simulated adversarial range manipulations and replay attacks. These findings confirm that decentralized PoL is feasible on embedded hardware with sub-meter accuracy, providing a concrete roadmap for turning theoretical models into deployable infrastructure.

Keywords: Proof-of-Location, Decentralized Systems, Prototype, Blockchain, Mesh network

CERCS: P170 - Computer science, numerical analysis, systems, control

Detsentraliseeritud asukoha tõendamise süsteemide praktilise teostuse suunas

Lühikokkuvõte:

Käesolev bakalaureusetöö lahendab detsentraliseeritud asukoha tõenduse (Proof-of-Location, PoL) süsteemide reaalse valideerimise puudumise küsimust, projekteerides ja eksperimentaalselt hinnates esimest füüsilist tunnistajate tsooni mudeli prototüüpi. Süsteem on välja töötatud soodsal laiatarbe-riistvaral ja see integreerib ESP32-DWM3000 ultralairiba (UWB) moodulid distantsi piiramiseks, Raspberry Pi 4 sõlmed B.A.T.M.A.N.-Adv võrgu loomiseks ning GoQuorum Istanbul BFT plokiahela andmete turvaliseks säilitamiseks. Protokolli kohandamiseks riistvaralistele piirangutele võetakse töös kasutusele tunnistaja algatatud kauguse mõõtmine ja krüptograafiline ploki räsi sidumine, et tõrjuda taasesitusrünnakuid füüsilisel kihil. Kokku 340 katsest koosneva testimise tulemused näitavad keskmist asukohaviga 0,18 m ja 100%-list heakskiitmise määra nominaalsetes tingimustes, tuvastades samas edukalt kõik simuleeritud rünnakud. Need leidud kinnitavad, et detsentraliseeritud PoL on teostatav madala maksumusega manussüsteemidel, pakkudes konkreetset tegevuskava teoreetiliste mudelite muutmiseks reaalseks infrastruktuuriks.

Võtmesõnad: Asukhatõendus, Detsentraliseeritud süsteemid, Prototüüp, Plokiahel, Silmusvõrk

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1. Introduction

Many digital services today rely on the assumption that a person or a device was really at a specific place at a specific time. Common examples are loyalty programs, geofenced content delivery, mobile voting, supply-chain tracking and the authentication of digital evidence in journalism [1, 2]. The usual source of location on consumer devices, the Global Navigation Satellite System (GNSS), was however not designed with adversarial users in mind. Spoofing tools are cheap and the device itself is the one that reports its own coordinates, so any service that simply trusts a self-reported position inherits the spoofing risk [3]. The growing use of generative artificial intelligence makes the problem even more pressing, because binding digital evidence to a verified place and time is one of the most direct ways to prove that something actually happened in the real world [2].

A Proof-of-Location (PoL) is a cryptographic certificate that addresses this problem. It allows a third-party verifier to be convinced that a prover was at a specific location at a specific time, based on the attestation of one or more witnesses [2]. Research on PoL has moved from the early centralized designs with a trusted location manager, through partially distributed protocols, to fully decentralized blockchain-based systems [4]. The most recent step in this trajectory is the unified architectural model of Brito et al. [2], which brings together ultra-wideband (UWB) ranging, distance bounding, mesh networking and permissioned Byzantine fault-tolerant consensus into one coherent protocol organized around the concept of a fault-tolerant witnessing zone. The model is, however, validated only by emulation on QEMU virtual machines, and the authors themselves leave the deployment on embedded devices with real radios as future work [2]. As a consequence, there is no published implementation of a decentralized PoL system that runs on real radios, in a real physical environment, with a real Byzantine fault-tolerant blockchain. Without such a deployment, the practical feasibility of the model cannot be confirmed and the engineering problems that appear only under real-world conditions remain hidden from view.

This thesis tries to fill that gap. The objective is to design, build and experimentally evaluate the first physical witnessing zone prototype that implements the decentralized PoL model of Brito et al. [2] on commodity low-cost embedded hardware, and to assess the real-world feasibility of the model. The contribution is fourfold. First, the thesis presents the first reported end-to-end physical prototype that integrates four ESP32-DWM3000 UWB witnesses, a Layer 2 B.A.T.M.A.N.-Adv mesh between four Raspberry Pi 4 nodes, a GoQuorum Istanbul Byzantine fault-tolerant blockchain and an independent verifier. Second, it adapts the abstract protocol to

packet-based UWB hardware through three concrete design choices: a witness-initiated Double-Sided Two-Way Ranging exchange, a global root-mean-square residual check that replaces the per-witness δ-test, and a cryptographic block-hash binding inserted into the UWB payload that defeats replay attacks at the physical layer. Third, the prototype is evaluated through 340 trials in a controlled indoor environment, under both honest and adversarial conditions, and produces a mean position error of $0.18\,\mathrm{m}$, a per-point acceptance rate of $100\,\%$ under honest operation and the successful detection of every simulated attack. Fourth, the thesis reports the engineering problems that emerged during the testing and proposes a concrete agenda for future work.

The remainder of the document is organized as follows. Chapter 2 reviews the literature on Proof-of-Location, traces the evolution of the protocols from the early centralized designs to the fully decentralized blockchain-based systems, presents the witnessing zone model of Brito et al. [2] in detail and identifies the open gap in real-world deployment that motivates the present work. Chapter 3 describes the methodology adopted in the thesis: the theoretical foundation of the protocol, including the roles, the distance bounding primitive together with its DS-TWR adaptation and the multilateration model, followed by the practical application development on the ESP32 and Raspberry Pi nodes and the independent verifier. Chapter 4 reports the experimental campaign in room 2018 of the Delta Building at the University of Tartu, defines the four test phases and the per-trial metrics, and discusses the results together with the practical limitations observed during the testing. Chapter 5 closes the thesis with the main conclusions and a set of concrete directions for future work. The complete source code, hardware setup notes and raw experimental data are provided in the accompanying GitHub repository [1] which is referenced from the relevant points in the text.

In the writing of this work, Anthropic's AI-based tool Claude Opus 4.7 was utilized for two purposes. First, the AI was used to improve formatting and to proofread for spelling and grammatical errors. The AI did not independently generate any substantive paragraphs; instead, it offered corrections to the existing text to enhance coherence. These rephrasing suggestions were used selectively, and the author made the final decision regarding all text. Second, the AI was used for code debugging in situations where it was necessary to understand the cause of an error message or unexpected behavior. In these instances, the existing code snippet and the error message were provided to the AI, and the resulting explanation was used to inform decisions

[1] https://github.com/tamor04/TamorTomsonPolThesis

for corrections. The AI was not used to create original textual or code-based components of the work, nor was it used to paraphrase the content of cited sources. All references to primary sources have been manually verified.

2. Background

This chapter places the present work into the context of the existing literature on Proof-of-Location (PoL). Section 2.1 explains why digital systems need a verifiable location and what problems appear when it is missing. Section 2.2 follows the historical development of PoL protocols, from the early designs based on a single trusted witness to the fully decentralized, blockchain-backed architectures. Section 2.3 describes in detail the decentralized witnessing-zone model of Brito et al. [2], which is the architectural basis of this thesis. Section 2.4 gives an overview of the enabling technologies, and Section 2.5 identifies the open gap in the current literature that motivates the work of this thesis.

2.1 Location Verification in Digital Systems

Many digital services today depend on the assumption that a user, a device or a piece of content was really present at a given place and time. Loyalty programs, geofenced content delivery, mobile voting, delivery and supply-chain verification, and the authentication of journalistic content all need a trustworthy location claim [1, 2]. The growth of generative artificial intelligence has made it even more important to be able to bind digital evidence to a specific time and place, because verifying the origin of content becomes increasingly difficult [2].

The main source of location on consumer devices, the Global Navigation Satellite System (GNSS), was not designed with adversarial users in mind. Cheap spoofing tools can produce fake satellite signals, and the device itself is the one that reports the resulting position. Any service that trusts a self-reported coordinate therefore inherits this spoofing risk [3]. Wi-Fi and cellular triangulation have the same weakness, and there is no cryptographic link between the reported position and the entity that claims it. Čapkun and Hubaux [3] solved part of this problem by combining distance bounding with verifiable multilateration, but the result is a positioning primitive, not a certificate of presence that can be transferred and verified later by a third party.

A *Proof-of-Location* is exactly such a certificate: a digital object that allows a verifier v, who may be absent from the event, to be convinced that a prover p was at location l and time t, based on the attestation of a set of witnesses W [2]. Two properties are commonly required in the literature. The first is *spatio-temporal soundness*: a prover who is not physically co-located with the witnesses at (l, t) must not be able to produce or forge a valid proof. The second is *completeness*: an honest prover and honest witnesses must always be able to convince an honest

verifier. Other properties such as privacy, non-transferability and incentive compatibility are protocol-specific and explain most of the differences between the designs reviewed below.

2.2 Evolution of Proof-of-Location protocols

Following the historical evolution described by Brito et al. [4], PoL protocols can be organized into three stages based on their trust assumptions.

2.2.1 Trusted and centralized architectures

The first proposal, by Waters and Felten [5], combined a trusted location manager with a tamper-resistant device to enforce physical-boundary policies, for example the supervision of borrowed equipment. Saroiu and Wolman [1] extended the list of use cases to loyalty rewards, location-restricted content and presence verification for voting, using Wi-Fi access points to produce the proofs. VeriPlace [6] added a privacy-aware architecture that distributed responsibilities between several trusted entities. SLVPGP [7] moved the trust onto tamper-resistant hardware modules. *I am Alice* [8] simplified proof generation by using the existing Wi-Fi infrastructure, and Akand et al. [9] proposed a centralized but formally secure scheme with provable resistance to geo-tampering. All these systems share one important limitation: they include a trusted central component, which is a single point of failure and also a single attractive target for collusion [4].

2.2.2 Partially distributed protocols

A second generation of protocols distributed the witnessing function across multiple peers. APPLAUS [10] used co-located Bluetooth devices that mutually generate location proofs, which are then stored in pseudonymized form on an untrusted location-proof server. STAMP [11] introduced a trust model based on entropy together with asymmetric cryptographic primitives, so that mutual co-located proofs can be made while still protecting the identity of individual signers. PROPS [12] continued in the same direction and used group-signature constructions to combine multi-witness proofs under a single anonymous credential. Both of these schemes, however, are still vulnerable to collusion between the prover and the witnesses. SPARSE [13] reduced this risk by removing the ability of the prover to choose its own witnesses, and used crowd density itself as a security primitive. Dupin et al. [14] replaced point-to-point trust with secure multi-party computation between semi-honest entities, but at the cost of a high computational overhead. These protocols clearly moved beyond the trusted-witness model, but they still depend on a centrally issued identity infrastructure and do not provide any tamper-evident global storage for the resulting proofs.

2.2.3 Fully decentralized and blockchain-based systems

The arrival of public ledgers finally removed the last centralized component. Amoretti et al. [15] combined short-range wireless attestations with a blockchain back-end, so that the proofs become tamper-evident and resistant to censorship without the need for a trusted server. Nasrulin et al. [16] formalized the spatio-temporal verification requirements and deployed a permissioned blockchain for supply-chain tracking. Wu et al. [17] added zero-knowledge proofs on top of the blockchain to support hierarchical and user-controlled disclosure (ZK-PoL). Nosouhi et al. [18] introduced incentive mechanisms that reward honest witnesses, and the FOAM whitepaper [19] described an actually deployed infrastructure where Zone Anchors (LPWAN radio beacons) are coordinated by Zone Authorities through Ethereum smart contracts, in order to maintain shared time and issue verifiable location claims. Pournaras's *Proof-of-Witness-Presence* [20] targets civic participation in smart-city environments. Even with this large design space, Brito et al. [2] observe that the previous contributions are still fragmented: each one solves a particular cryptographic, networking or consensus problem, but none of them integrates everything into one complete, end-to-end decentralized PoL infrastructure that is demonstrated outside of an emulator.

2.3 The Decentralized Witnessing-Zone Model

Brito et al. [2] bring together the previously scattered techniques into a single architectural model, which is the direct basis of the present work. The entire protocol is captured by three roles and one structural primitive, the *witnessing zone*. A *prover* p is the entity whose presence has to be attested. A *witness* w_i is a stationary node with a known public key, which observes provers and contributes a partial attestation. A *verifier* v is any third party that later evaluates a proof against the public state of the system, possibly long after the actual event [2].

A zone Z_z is a region of space–time inside which a finite set of witnesses can communicate reliably and agree on a logical clock. The zone is defined through spatial, temporal and connectivity constraints over the witness-to-witness channel C_w, while a separate witness-to-prover channel C_p is reserved for distance bounding and attestation [2]. Multiple zones can overlap and form a continuous lattice; the smaller the zone, the better the spatial precision of the resulting proof, but the higher the deployment density required. In Figure 1, a single witnessing zone consisting of four witnesses, a prover inside the zone and a third party verifier outside of it is shown.

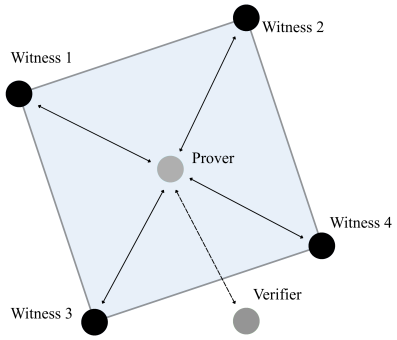

Figure 1. A single decentralized witnessing-zone, it being shown colored in gray.

Inside a zone, the witnesses agree on the order of events by means of a Byzantine fault-tolerant (BFT) consensus mechanism with deterministic finality. This produces a hash-linked blockchain $B = \{b_0, b_1, \dots\}$ that is local to the zone [2, 21]. Tolerating f Byzantine witnesses requires $n \geq 3f + 1$ honest participants. In two-dimensional zones this gives a minimum of four coplanar witnesses, which also satisfies the geometric requirement of multilateration with one-fault tolerance at the same time [2]. For each round, the prover engages in a distance-bounding interaction $d_{b_i} = DB(w_i, p)$ with every witness, and collects partial signatures over a block candidate b_i that records the prover's identity, the distance bounds and the metadata. The proof is then assembled as

$$\pi = (b_i, \sigma), \quad \sigma = \text{Aggregate}(\text{sig}_{w_1}, \dots, \text{sig}_{w_{n_b}}),$$

where σ is a collective signature that requires at least n_b out of n witnesses. A verifier accepts π if and only if σ is valid over b_i and the multilaterated position falls inside the zone [2].

It is important to note that Brito et al. [2] validated this model only through emulation. They used QEMU virtual machines running OpenWrt with the B.A.T.M.A.N.-Adv kernel module and an Ethereum-style Proof-of-Authority chain. Real radios, real distance bounding and real-world propagation effects were explicitly left as future work. The authors themselves state that the next steps are to deploy the system on embedded devices that have integrated distance-bounding hardware, in order to validate the real-world feasibility of the model [2]. This lack of real-world deployment and testing is the open challenge that the present thesis tries to address.

2.4 Enabling Technologies

A practical realization of the witnessing-zone model is built on four technology layers. The first one is **Ultra-wideband (UWB)**, a short-range radio technology that uses nanosecond

pulses spread over a bandwidth of at least 500 MHz. This gives sub-decimeter time-of-flight resolution and strong resistance to narrow-band multipath. The IEEE 802.15.4z amendment [22] standardizes the enhanced UWB physical layers and their ranging procedures, including single-sided and double-sided two-way ranging (SS-TWR and DS-TWR) and cryptographic constructs that protect ranging integrity against external attackers. Among consumer-grade radios, UWB is currently the only mainstream option that provides the temporal resolution needed for sub-meter PoL.

The second layer is **distance bounding**, introduced by Brands and Chaum [23]. It allows a verifier to establish a cryptographic upper bound on the physical distance to a prover, by sending unpredictable single-bit challenges and measuring the round-trip time of the prover's commitment-bound responses. The speed of light then gives a hard upper bound on the claimed proximity. Rasmussen and Čapkun [24] demonstrated the first radio-frequency realization of this idea on custom hardware. On commodity radios, the classical rapid bit-exchange is difficult to implement directly because of the medium-access overhead. For this reason, the IEEE 802.15.4z DS-TWR procedure [22] is widely used as an engineering approximation.

The third layer is **wireless mesh networking**. This is needed because a witnessing zone has no privileged node and no managed infrastructure, so the witnesses must self-organize into a fully connected mesh. Several physical layers can support upper-layer mesh protocols [2]. For Wi-Fi, B.A.T.M.A.N.-Adv [25] operates at Layer 2 and exposes a flat, Ethernet-like interface that isolates the consensus and ranging code from the routing details. The same protocol was used in the emulated zone of Brito et al. [2].

The fourth and final layer is **permissioned blockchain consensus**. This choice is appropriate inside a zone because the validator set is bounded and known in advance. Practical Byzantine Fault Tolerance [21] and its modern descendants provide the $n \geq 3f + 1$ safety guarantee with sub-second finality. Istanbul BFT (IBFT), implemented for example by GoQuorum, adds smart-contract execution and a validator set that maps naturally onto the witness set of a zone, which realizes the programmable witnessing zone described in the decentralized model [2].

2.5 Conclusion

PoL has matured over the years, starting from the trusted-witness designs, then moving to partially distributed protocols with privacy-preserving multi-witness signatures, and finally arriving at fully decentralized blockchain-based systems. Brito et al. [2] bring this trajectory together

into a single witnessing-zone architecture and demonstrate its viability through emulation. The technologies that are needed to deploy this architecture: UWB ranging, distance bounding, Layer-2 mesh networking and permissioned BFT consensus, are all individually mature. However, no work in the existing literature reports an end-to-end physical implementation that integrates all of them into a real witnessing zone. The current state of the art therefore still misses the most important step, which is real-world validation. The remainder of this thesis tries to fill this gap. The theoretical foundation and the engineering of the first witnessing zone prototype are presented in Chapter 3 and the experiments conducted with that prototype with the results gained from the test are described in Chapter 4.

3. Methodology

3.1 Introduction

As described in Chapter 2, the decentralized Proof-of-Location (PoL) architecture proposed by Brito et al. [2] brings together UWB ranging, distance bounding, Layer-2 mesh networking and permissioned BFT consensus into one unified model. The same authors, however, validated this architecture only through emulation on QEMU virtual machines, and they explicitly leave the deployment on real embedded devices with real radios as future work [2]. The lack of real-world deployment and testing is, at the moment, the most important open challenge in the field.

The goal of this thesis is to address exactly this challenge. To do that, this chapter describes the design, the construction and the validation procedure of the first physical witnessing zone prototype that implements the decentralized PoL architecture of Brito et al. [2]. Unlike the original work, which is based only on virtual machines, the present thesis adopts a *prototype-driven* methodology: every component of the protocol is built on real, low-cost embedded hardware and is tested under real radio-frequency, networking and timing conditions.

The chapter is organized in two parts. Part 1, the *Theoretical Foundation* (Section 3.2), explains the concept behind Proof-of-Location, the protocol roles and the message flow, the distance bounding primitive together with the Double-Sided Two-Way Ranging (DS-TWR) adaptation, the multilateration model that is used for position estimation, and the proof structure together with its verification model. Sequence diagrams and mathematical models are included to support the explanation. Part 2, the *Application Development and Implementation* (Sections 3.3–3.6), covers the practical side. It introduces the layered system architecture and the network topology (Section 3.3), the required hardware setup (Section 3.4), the software implementation on the ESP32 microcontrollers and the Raspberry Pi nodes (Section 3.5), and finally the independent verifier (Section 3.6), Section 3.7 closes the chapter and connects it to the experimentation of the prototype in Chapter 4.

3.2 Theoretical Foundation

3.2.1 Proof-of-Location Protocol Overview

A Proof-of-Location protocol produces a verifiable and tamper-resistant certificate that a digital entity, the *prover*, was physically present at a specific place and at a specific time. The decentralized version of this idea distributes the trust across three logical roles. The prototype reproduces all of these roles faithfully in hardware:

Prover (p). A mobile entity that wants to prove its presence inside a witnessing zone and obtain a location proof.

Witness (w_i). A spatially fixed node with a known position (x_i, y_i) that participates in the distance bounding interaction with the prover and contributes a signed attestation to a shared ledger. One zone contains n such witnesses.

Verifier (v). An independent third party that, given a proof, decides whether the presence claim of the prover is valid or not. The verifier is honest, but it does not trust the witnesses or the prover a priori.

A *witnessing zone* Z_z is a region of space-time inside which the witnesses can communicate reliably and remain logically synchronized. Following [2], a zone is formally defined as

$$Z_z = \left\{ w_i \;\middle|\; \forall w_j \in Z_z : (w_i, w_j) \in C_w \;\wedge\; d(w_i, w_j) \leq R_{\text{max}} \;\wedge\; |t_i - t_j| \leq \epsilon \right\}, \tag{1}$$

where $d(\cdot, \cdot)$ is the Euclidean distance, R_{max} is the maximum reliable communication range, ϵ is the temporal precision bound, and C_w denotes the witness-to-witness communication channel. A second channel C_p carries the witness-to-prover messages that are used for distance bounding and proof delivery.

The protocol, as implemented in the prototype, proceeds in five logical steps:

1. *Entry.* The prover enters zone Z_z and announces its public identity pk_p to the witnesses through C_p.
2. *Distance bounding.* Each witness w_i runs a distance bounding interaction with p and obtains an upper bound $db_i = \text{DB}(w_i, p)$ on their physical separation.
3. *Commitment.* Each witness signs a structured record that contains pk_p, db_i and the timing metadata, and commits this record to the witness blockchain through a Byzantine fault-tolerant consensus protocol.
4. *Collection.* The prover monitors the blockchain and collects all witness commitments $\{\sigma_i\}_{i=1}^{n_b}$ that refer to itself inside a bounded time window.
5. *Assembly.* The prover compiles the final location proof

$$\pi = (b_i, \sigma_1, \ldots, \sigma_{n_b}), \tag{2}$$

where b_i is the block identifier that orders and time-stamps the attestations, and submits π to a verifier.

Trust in the system emerges from the combination of three independent properties: the spatial soundness of distance bounding (a remote prover cannot appear closer than the speed of light allows), the Byzantine fault-tolerant agreement of $n \geq 3f + 1$ witnesses (at most f of them may be malicious), and the deterministic finality of the blockchain (the events are totally ordered and tamper-evident). For the prototype, $n = 4$ and $f = 1$, which is the minimum BFT configuration that can tolerate one faulty witness.

3.2.2 Distance Bounding and Two-Way Ranging

Distance bounding is an abstract cryptographic primitive $\mathrm{DB}(w_i, p)$ that allows a verifier to obtain a strict upper bound on the physical distance to a prover. This is done by measuring the round-trip time of a fast challenge–response exchange. The classical construction of Brands and Chaum [23] consists of k rounds of single-bit challenges and single-bit responses. The prover commits cryptographically to its responses before the ranging phase, so that an adversary cannot precompute valid bits. The verifier then upper-bounds the distance by

$$d \leq \frac{c \cdot (t_{\text{rx}} - t_{\text{tx}} - t_{\text{proc}})}{2}, \tag{3}$$

where c is the speed of light, t_{tx} and t_{rx} are the transmission and reception times of the challenge–response pair, and t_{proc} is the internal processing time of the prover. Rasmussen and Čapkun later showed that an RF realization of this primitive is feasible only on specialized hardware that is able to do nanosecond-scale signal turnaround [24].

The Qorvo DWM3000 transceiver used in the prototype implements the IEEE 802.15.4z amendment for ultra-wideband ranging [22], which standardizes packet-level exchanges instead of bit-level ones. Because every UWB frame carries a MAC header, a preamble and a CRC, the processing delay between reception and transmission is in the order of hundreds of microseconds and not nanoseconds. For this reason, the classical single-bit distance bounding cannot be executed on these radios. The prototype therefore adopts *Double-Sided Two-Way Ranging (DS-TWR)* [26] as a practical realization of the distance bounding primitive. DS-TWR uses three frames: Poll, Response and Final, and four timestamps that are measured on both endpoints. With $T^A_{\text{round}}, T^A_{\text{reply}}$ measured at the witness and $T^B_{\text{round}}, T^B_{\text{reply}}$ measured at the prover, the time of flight is estimated as

$$T_f = \frac{T^A_{\text{round}} \cdot T^B_{\text{round}} - T^A_{\text{reply}} \cdot T^B_{\text{reply}}}{T^A_{\text{round}} + T^B_{\text{round}} + T^A_{\text{reply}} + T^B_{\text{reply}}}, \tag{4}$$

which gives a first-order compensation of the relative clock drift between the unsynchronized endpoints [26] and produces the distance estimate $d_i = c \cdot T_f$.

Because DS-TWR does not enforce nanosecond-scale processing constraints, it cannot, on its own, prevent a remote adversary from relaying the messages and appearing close. To recover a form of liveness on packet-level UWB hardware, this thesis introduces an additional binding mechanism: each witness Raspberry Pi periodically injects the hash of the latest finalized block into its UWB transceiver. The witness includes this hash in the Poll message, and the prover must echo it back inside the Response payload. A witness rejects any range measurement whose echoed hash does not match the latest or the immediately preceding block hash, because such a hash could not have been known before consensus on the witness blockchain. This scheme binds every accepted distance bound to a specific point of the blockchain history. It mitigates replay attacks, at the cost of relying on the freshness of the witness ledger instead of relying on photon-speed processing.

3.2.3 Multilateration and Position Estimation

Given $n \geq 3$ distance bounds $d_1, \ldots, d_n$ and known witness coordinates $\mathbf{c}_i = (x_i, y_i)$, the position of the prover $\mathbf{x} = (x, y)$ is recovered through *multilateration* [3]. Each distance measurement defines the residual

$$r_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}_i\| - d_i, \tag{5}$$

Čapkun and Hubaux [3] write the residual with the opposite sign, as $f_i(\mathbf{x}) = d_i - \|\mathbf{x} - \mathbf{c}_i\|$. The two forms are equivalent under the squared sum that is minimized in the next step, so the choice is only a matter of convention. The form used here matches the implementation in `verifier_script.py`. and the position is estimated as the Minimum Mean Square Estimate (MMSE)

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathbb{R}^2} \sum_{i=1}^{n} r_i(\mathbf{x})^2, \tag{6}$$

which is a non-linear least-squares problem. The prototype solves it with the Levenberg–Marquardt algorithm [27], as implemented in the `scipy.optimize.least_squares` routine of the SciPy scientific computing library [28]. The minimum number of witnesses needed to obtain an unambiguous 2D solution is three. The prototype uses four witnesses, in order to also satisfy the Byzantine fault tolerance bound $n \geq 3f + 1$ for $f = 1$.

The minimized residual sum of squares can also be used as a figure of merit for the geometric consistency of the solution. Čapkun and Hubaux [3] apply a per-verifier δ-*test* on the estimated position $\hat{\mathbf{x}}$:

$$\forall i \in \{1, \ldots, n\} : \ \left| d_i - \|\hat{\mathbf{x}} - \mathbf{c}_i\| \right| \leq \delta, \tag{7}$$

which rejects the proof as soon as the residual of any single witness exceeds the tolerance δ. The prototype adopts a different criterion: instead of bounding each residual individually, it bounds the aggregate Root-Mean-Square (RMS) residual

$$\rho_{\text{RMS}}(\hat{\mathbf{x}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} r_i(\hat{\mathbf{x}})^2}, \tag{8}$$

and rejects any proof for which $\rho_{\text{RMS}}(\hat{\mathbf{x}}) > \tau_{\text{RMS}}$ with $\tau_{\text{RMS}} = 1.5\,\text{m}$. This threshold value is tuned to the noise floor observed in the testing environment. The RMS criterion is weaker than the per-verifier δ-test, because a single bad witness can be averaged out by three good ones. It is, however, more tolerant of the normal ranging noise of the DWM3000 modules, which can produce per-witness errors close to δ even when no attack is taking place. The trade-off between the two criteria is examined empirically in Chapter 4.

A second geometric check, also from the verifiable multilateration of Čapkun and Hubaux [3] and later refined by Chiang, Haas and Hu [29], is applied during independent verification. This check is purely geometric and operates on the single estimated position $\hat{\mathbf{x}}$ produced by the MMSE step. For every triple of witnesses (w_i, w_j, w_k), the verification triangle $\triangle(\mathbf{c}_i, \mathbf{c}_j, \mathbf{c}_k)$ is formed from their known coordinates. The proof is accepted if

$$\exists\,(i,j,k):\ \hat{\mathbf{x}} \in \triangle(\mathbf{c}_i, \mathbf{c}_j, \mathbf{c}_k), \tag{9}$$

that is, if at least one such triangle contains the estimated position. The residual check of Equation (8) and the triangle test of Equation (9) together catch coordinated tampering: a malicious witness who inflates its distance pulls the estimate either to a position with a large residual, which fails the first test, or outside every verification triangle, which fails the second test. Neither test alone is sufficient.

3.2.4 Proof Structure and Verification Model

The location proof is a structured object

$$\pi = \big(b_i, \{\sigma_j\}_{j=1}^{n_b}, \mathbf{x}, \{d_j\}, \{\mathbf{c}_j\}, \{t_j\}\big) \cdot \text{Sign}_{\text{sk}_p}, \tag{10}$$

where b_i is the latest blockchain block that is included in the proof, σ_j are the per-witness signatures of the distance claims retrieved from the chain, $\mathbf{x}$ is the estimated position, d_j and $\mathbf{c}_j$ are the bounds and the coordinates used in multilateration, t_j are the timestamps of the witness commitments, and $\text{Sign}_{\text{sk}_p}$ is an Elliptic-Curve Digital Signature Algorithm (ECDSA) signature produced by the prover over the entire structure, in order to bind the assembled proof to its identity.

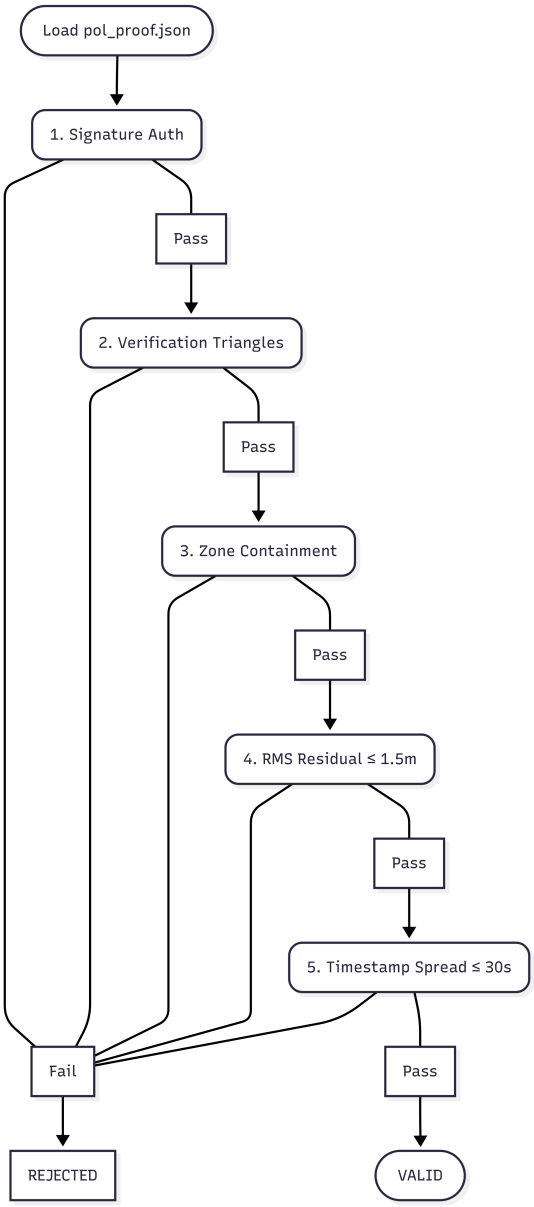

Figure 2. Verification pipeline

The verifier accepts π only if all of the following five predicates evaluate to true:

1. **Signature authentication.** The ECDSA signature of the prover correctly recovers the address embedded in π. This confirms that the proof has not been modified after assembly.
2. **Geometric consistency.** The estimated position $\hat{\mathbf{x}}$ lies inside at least one verification triangle $\triangle(\mathbf{c}_i, \mathbf{c}_j, \mathbf{c}_k)$ formed by a triple of witness coordinates, as defined in Equation (9) [3, 29].
3. **Zone containment.** $\hat{\mathbf{x}}$ lies inside the convex hull of the known witness coordinates, which defines the physical zone.

4. **RMS residual.** $\rho_{\text{RMS}}(\hat{\mathbf{x}}) \leq \tau_{\text{RMS}}$, as defined in Equation (8).

5. **Temporal freshness.** The spread of the witness commitment timestamps $\max_j t_j - \min_j t_j$ is bounded by a configurable maximum $\tau_t = 30\,\text{s}$.

A failure on any single predicate causes the immediate rejection of the proof. Figure 2 shows the verification pipeline as a flowchart.

3.3 System Architecture

Now that the theoretical foundation has been described, the rest of the chapter explains how this theory is actually realized in hardware and software. This is the practical part of the methodology: the system architecture, the hardware setup, the software implementation, the independent verifier, and the experimental procedure.

3.3.1 Architecture Overview

The prototype is organized as a stack of four functional layers, which together map the protocol of Section 3.2.1 onto commodity hardware:

Physical layer. Provides the ultra-wideband ranging between the prover and each witness. It is built from five ESP32 microcontrollers, each paired with a Qorvo DWM3000 UWB module: one for the prover and four for the witnesses.

Bridge layer. Converts the ranging events into authenticated blockchain transactions. Each witness runs a Python script on a Raspberry Pi 4. This script reads the UWB measurements over USB, validates them against the latest block hash, signs them with the witness keypair and submits them to the consensus layer.

Consensus layer. Records the distance claims in an append-only ledger. A four-node Quorum [30] blockchain runs under Istanbul Byzantine Fault Tolerance (IBFT) [31], is deployed in Docker containers, and operates over an isolated wireless mesh.

Application layer. Aggregates the witness claims into a location proof and exposes it for verification. A Python script on the prover Raspberry Pi monitors the blockchain, performs the multilateration, applies the rejection checks and exports the serialized proof. An independent verifier validates this proof offline.

Figure 3 shows the four-layer architecture of the prototype and the main data flows between the layers, labelled Physical, Bridge, Consensus and Application. In the physical layer, the prover ESP32 and the four witness ESP32 boards exchange UWB DS-TWR ranging frames; each

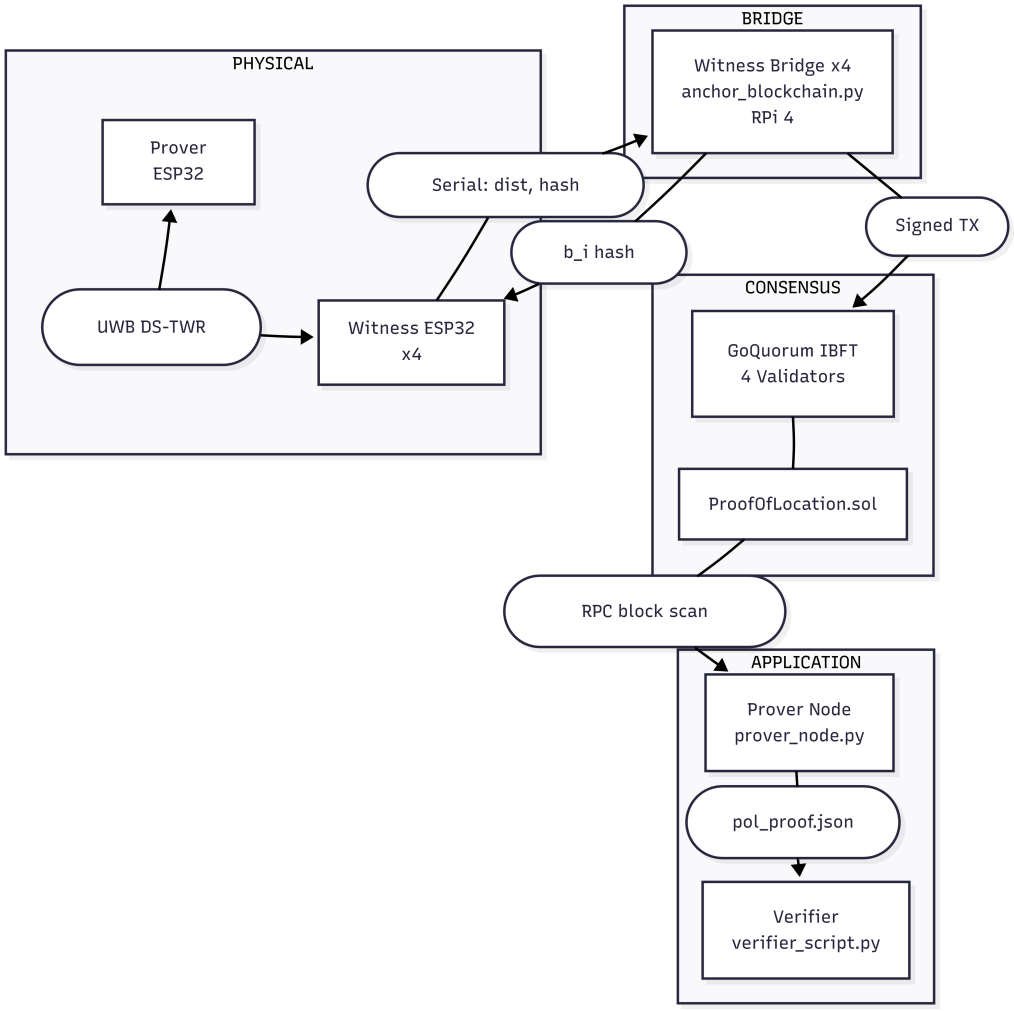

Figure 3. Architecture and data flow of the system

witness then sends the measured distance and commitment hash to its paired Raspberry Pi over a serial link, while the Raspberry Pi pushes the latest block hash b_i back to the witness, so that every ranging round is bound to a recent state of the ledger. The bridge layer signs these statements and submits them as transactions to a four-validator GoQuorum IBFT network, where they are recorded through the `ProofOfLocation.sol` smart contract, and the application layer then reads the ledger over RPC, aggregates the claims into `pol_proof.json`, and passes it to the independent verifier, which is logically outside the witnessing zone and does not need to trust any other component of the system.

Three architectural choices need to be explicitly justified. First, four witnesses are deployed because four is the minimum number that satisfies the BFT bound $n \geq 3f + 1$ with $f = 1$. This value also matches the one used in the original analysis [2], and it supports the verification triangles check, which needs at least three honest witnesses. Second, Quorum with IBFT was

selected because the protocol requires deterministic finality, not probabilistic finality. IBFT provides this finality with a constant-size validator set and pluggable authority lists, which makes a four-node permissioned deployment self-contained. Third, the B.A.T.M.A.N. Advanced (batman-adv) Layer-2 mesh protocol [25] is chosen as the underlying network as Brito et al. also used it in their emulation [2], because it works as a virtual Ethernet bridge that is transparent to the blockchain stack, and because it does not need any infrastructure such as access points or DHCP servers.

3.3.2 Network Topology

The witness Raspberry Pis form an ad-hoc Wi-Fi mesh on the `10.0.0.0/24` subnet. Each node has a static IP in the range `10.0.0.1`–`10.0.0.4`. The batman-adv kernel module bridges all four wireless interfaces into a single virtual interface `bat0`. On top of this interface, the Quorum nodes communicate using their statically configured `static-nodes.toml` topology. Every blockchain message therefore passes through the mesh, which makes sure that the consensus behavior observed during the evaluation also includes all physical-layer disturbances such as multipath fading and packet loss.

The prover Raspberry Pi is intentionally kept outside the mesh, and it connects to one of the witnesses over Wi-Fi or Ethernet purely as a read-only Remote Procedure Call (RPC) client. It does not participate in consensus, does not relay any traffic, and has neither the genesis file nor the validator keys. This separation makes sure that the prover cannot influence which witness commitments are admitted to the chain. The independent verifier is even more strongly isolated: it runs on a laptop that has no network connectivity to either the mesh or the prover, and it operates exclusively on a serialized JSON proof obtained out of band.

3.4 Hardware Setup

3.4.1 Components and Bill of Materials

The complete bill of materials for the prototype is summarized in Table 1. Each Raspberry Pi 4 carries one ESP32 development board, which in turn drives one DWM3000 module over the SPI interface of the ESP32. The four witness platforms are all identical. The prover platform is identical to a witness as well, except for its firmware role and the way its Raspberry Pi connects to the mesh.

Table 1. Bill of materials for the prototype.

Component	Quantity	Role
Raspberry Pi 4 Model B (4 GB)	5	Bridge / aggregation
ESP32-WROOM development board	5	UWB controller
Qorvo DWM3000EVB module	5	UWB transceiver
USB-A to micro-USB cable	5	ESP32 power & serial
Jumper wires, breadboards	—	SPI wiring

Table 2. SPI pin assignment between the ESP32 and the DWM3000 module.

DWM3000 pin	ESP32 GPIO	Function
VCC	3.3 V	Power supply
GND	GND	Ground
SCK	GPIO18	SPI clock
MOSI	GPIO23	SPI data (ESP32 $\rightarrow$ DWM3000)
MISO	GPIO19	SPI data (DWM3000 $\rightarrow$ ESP32)
CS	GPIO4	SPI chip select
RST	GPIO27	Hardware reset
IRQ	GPIO34	Interrupt line

The SPI pin assignment between the ESP32 and the DWM3000 module is identical for every platform and is given in Table 2. The module strictly requires a 3.3 V supply and does not tolerate any higher voltage.

3.4.2 Physical Assembly and Deployment

Each witness unit consists of an ESP32 board wired to a DWM3000 module on a breadboard. This unit is connected through micro-USB to its Raspberry Pi, which itself is powered by a dedicated 5 V/3 A supply. Figure 4 shows one of the assembled witness nodes. The witnesses are deployed at the specific locations in room 2018 of the Delta Building at the University of Tartu. The prover is placed at different marked test points in the room and outside of it. The coordinate system and the layout of the room are discussed in more detail in Section 4.2.1.

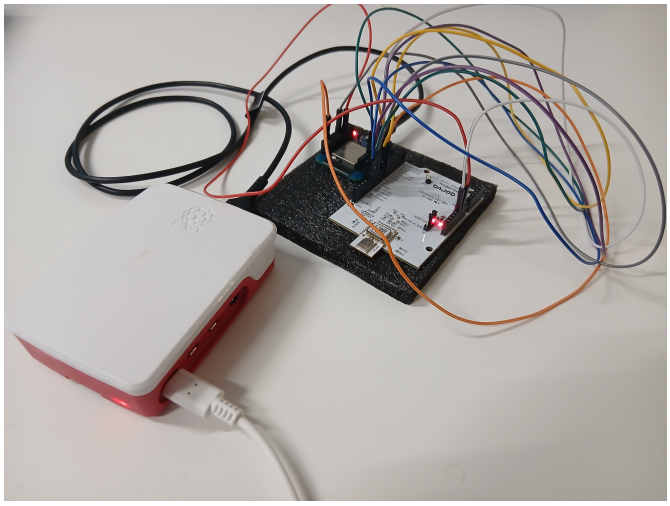

Figure 4. One assembled witness node: ESP32 board, DWM3000 UWB module, jumper wiring and Raspberry Pi 4 controller.

3.5 Software Implementation

3.5.1 UWB Firmware on the ESP32 + DWM3000

The ESP32 firmware drives the DWM3000 in the IEEE 802.15.4z UWB band [22], using channel 5, a 128-symbol preamble and a data rate of $6.8\,\mathrm{Mb/s}$. The driver part of the firmware is derived from the open-source ESP32+DWM3000 indoor positioning project of Waran [32]. It is reused mainly for the SPI initialization, the register access and the basic configuration of the radio. The protocol logic on top of the driver, however, is reimplemented from scratch.

Two functional roles share the same driver: a responder role on the prover ESP32 and an initiator role on each witness ESP32. Unlike Waran's prover-initiated indoor positioning, the prototype reverses the role of the initiator, so that a witness starts every ranging cycle. This reversal is essential for the security model, because a prover-initiated round would allow a remote adversary to choose the timing of the exchange and to pre-commit favorable responses. With witness-initiated rounds, the prover must always answer to a challenge that is controlled by the witness.

A complete DS-TWR cycle between a witness and the prover consists of three frames:

1. **Poll.** The witness transmits a Poll frame that contains its UWB identifier and the current blockchain hash h_b.

2. **Response.** The prover replies with a Response frame that contains a cryptographic commitment Commit $=$ SHA-256(p_i) to a fresh random nonce p_i, together with its own preliminary timestamps.

3. **Final.** The witness transmits a Final frame. After this, the prover sends a Report that opens the commitment (by revealing p_i) and includes the four timing values that the witness needs in order to compute Equation (4). The witness then verifies that SHA-256$(p_i) =$ Commit before accepting the measurement.

If any frame is lost or if the commit does not verify, the witness aborts the cycle, prints a diagnostic message and waits for one ranging interval before retrying. Because all four witness's UWB modules share the same UWB channel, the firmware uses a static Time Division Multiple Access (TDMA) schedule based on the device's identifier: they stagger their initial transmission by an identifier-dependent delay, so that simultaneous Polls do not collide on air. Once a ranging cycle is completed, ESP32 emits a single JSON record over USB serial to its Raspberry Pi. This record contains the UWB identifier, the computed distance in centimeters, the four DS-TWR timing values, a local timestamp in milliseconds and the echoed blockchain hash h_b. The hash is later compared with the latest finalized block on the bridge layer in order to detect replay attempts, as described in Section 3.5.2.

3.5.2 Witness Bridge Software

Each witness Raspberry Pi runs the Python script `anchor_blockchain.py`, which acts as the bridge between the UWB serial stream and the blockchain. Its responsibilities are executed in a single asynchronous loop:

1. read one JSON record from the serial port of the ESP32;

2. validate that the reported distance is positive and below the maximum range $R_{\text{max}} = 20\,\text{m}$;

3. compare the echoed blockchain hash h_b against the latest and the immediately preceding block hashes obtained through the local Quorum RPC interface, and discard the record if both comparisons fail (replay detection);

4. build a transaction that calls the `recordDistanceClaim` function of the `ProofOfLocation` smart contract, sign it with the locally generated secp256k1 keypair of the witness and submit it to the network;

5. wait for the transaction receipt before continuing, so that the claim is finalized by IBFT before the next measurement is processed.

A rate limit of three seconds between transactions throttles the ranging cycle to a sustainable rate over the wireless mesh, and prevents flooding the consensus layer during dense ranging bursts. The bridge also injects the latest block hash back into the ESP32 once per second, through a JSON command on the serial channel. This way, the witness UWB always has fresh material to send to the prover in the next Poll frame.

The bridge also accepts three command-line flags that enable controlled adversarial behavior for the experiments described in Section 4.2: `--inject-delay-ms` delays each transaction submission, `--corrupt-range-offset` adds a constant offset to the reported distance, and `--stale-timestamp` feeds the ESP32 a block hash from $\Delta_b = 10$ blocks in the past. The first two flags simulate synchronization stress and a malicious range manipulation. The third one triggers the anti-replay defense on every other witness.

3.5.3 Blockchain Layer

The consensus layer is a Quorum permissioned Ethereum network [30] that operates under Istanbul Byzantine Fault Tolerance [31]. IBFT is a variant of the practical Byzantine Fault Tolerance algorithm of Castro and Liskov [21], adapted to the Ethereum execution environment. Four validator nodes form the validator set, which is the minimum number that supports the $n \geq 3f + 1$ requirement for $f = 1$. The genesis configuration assigns chain identifier 10, a block period of 10 seconds, a fixed gas price of zero, and an extra-data field that hard-codes the four authorized validator addresses. A static-nodes manifest fixes the peer topology to the mesh IPs `10.0.0.1`–`10.0.0.4`, so that no peer discovery or external bootnode is involved.

A single Solidity smart contract, `ProofOfLocation.sol`, defines an append-only data type and the function

```
function recordDistanceClaim(
    address proverAddress,
    uint256 tRoundB,
    uint256 tReplyB,
    uint256 distCmX100
) external;
```

which stores the tuple $(w_i,\, p,\, T^B_{\text{round}},\, T^B_{\text{reply}},\, d_i,\, t_i)$ in the state of the contract and emits an event so that the prover can subscribe to new claims. The function itself performs no validation: all the validation is enforced at the bridge layer, in order to keep the contract minimal and gas-efficient.

One practical limitation of the prototype is that the isolated mesh has no path to an external time reference, so the Network Time Protocol cannot be used. As a consequence, the wall-clock times reported in the `timestamp` field of each block are set manually before deployment, and are subject to drift between witnesses over long runs. The temporal freshness predicate of the verification model (Section 3.2.4) tolerates this drift by bounding only the relative spread between witnesses, instead of the absolute time.

3.5.4 Prover Aggregation Logic

The prover Raspberry Pi runs the script `prover_node.py`, which orchestrates the assembly of the proof. On startup, the prover generates a fresh secp256k1 keypair, computes the corresponding address and connects to the Quorum RPC endpoint of one of the witnesses as a read-only client.

The prover then enters a loop. For every new block, it scans the block's transactions and looks for invocations of `recordDistanceClaim` that are addressed to the prover. Each matching transaction is decoded into a distance claim, indexed by the witness identifier embedded in the address, and stored in a buffer. Claims older than $\Delta_w = 4$ blocks or $\Delta_t = 30\,\text{s}$ are expired, so that the proofs are always built from a temporally coherent set of claims. When the buffer contains at least $n_b = 3$ distinct witness claims, the prover performs the following steps:

1. sort the claims by UWB module index and extract the distance vector (d_i) and the coordinate vector $(\mathbf{c}_i)$;
2. solve the multilateration problem of Equation (6) with `scipy.optimize.least_squares`, in order to obtain $\hat{\mathbf{x}}$;
3. compute the RMS residual of Equation (8) and the timestamp spread $\max_i t_i - \min_i t_i$;
4. apply the rejection rules: the position must lie inside the convex hull defined by $\{\mathbf{c}_i\}$, the RMS residual must not exceed τ_{RMS}, and the timestamp spread must not exceed τ_t;
5. assemble the proof of Equation (10), sign it with the private key of the prover and write `pol_proof.json` to disk;
6. clear the buffer and resume monitoring.

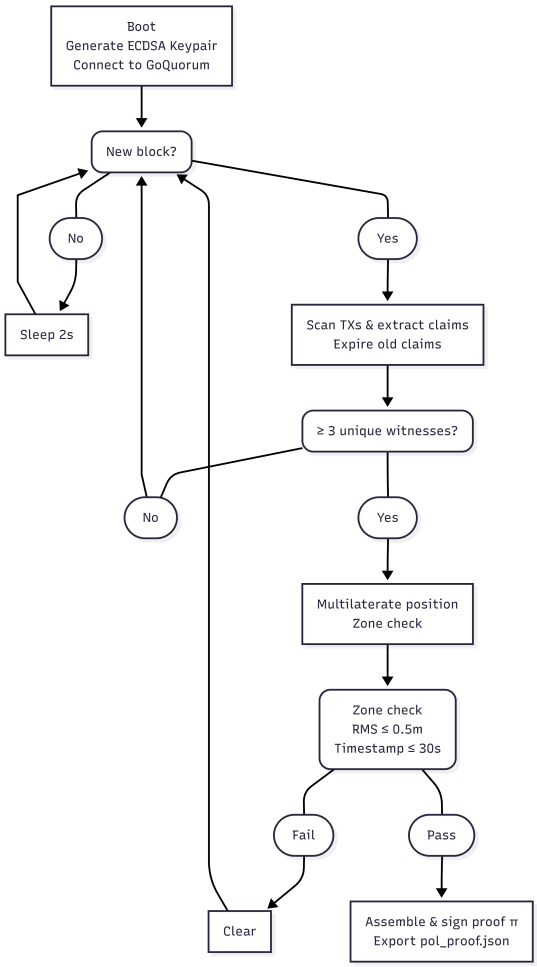

Figure 5. Pipeline of prover_node.py script

A rejection at any step is logged together with its reason, `outside_zone`, `rms_exceeded`, `timestamp_spread` or `quorum_not_met`, using the metrics defined in Section 4.2.3. This way, the acceptance rate can be analyzed per scenario in Chapter 4. Figure 5 summarizes this proof pipeline.

3.6 Independent Verifier

The verifier is implemented as the standalone Python script `verifier_script.py`. It accepts a single `pol_proof.json` file produced by the prover and runs in a process that has no access to the mesh, to the blockchain or to the prover. The script applies the five predicates of Section 3.2.4 one by one, in order:

1. Recover the address of the prover from the ECDSA signature over the proof body, and compare it to the address embedded in the proof.

2. Apply the verification triangles check, which originates in Čapkun and Hubaux [3] and was extended by Chiang, Haas and Hu [29]: build the triangle formed by every triple of witness coordinates, and check whether the estimated position $\hat{\mathbf{x}}$ lies inside at least one of them. Reject the proof if no triangle contains the position.
3. Check whether the estimated position lies inside the pre-configured polygonal zone (the convex hull of the witness coordinates).
4. Compute the RMS residual from the carried distances and coordinates, and compare it with τ_{RMS}.
5. Compute the timestamp spread and compare it with τ_t.

The first failing predicate terminates the verification with a labeled rejection. If all five predicates succeed, the verifier prints the estimated coordinates and the final `[VALID]` marker. This strict left-to-right evaluation order makes sure that each rejection reason is unambiguous in the experimental logs.

3.7 Conclusion

The methodology adopted in this thesis turns the abstract Proof-of-Location architecture of Brito et al. [2] into a complete prototype that covers, end to end, the ultra-wideband physical layer, an isolated wireless mesh, a permissioned IBFT blockchain, a proof assembly component and an independent verifier. Section 3.2 establishes the theoretical foundation that guides the engineering work: the protocol roles, the distance bounding primitive together with its DS-TWR adaptation for packet-level UWB hardware, the multilateration model used for position estimation, and the proof structure together with the five-predicate verification model. Sections 3.3 through 3.6 describe the application development and implementation: the layered system architecture, the hardware setup, the software implementation and the independent verifier. Figure 6 shows the the system's workflow starting from the communication between prover and witness ESP32 and ending with the verification of the location proof.

The original design contributions of this thesis: the anti-replay block-hash binding over UWB, the RMS residual threshold of $1.5\,\mathrm{m}$ as an alternative to the per-witness δ-test, the witness-initiated DS-TWR role reversal and the five-predicate verification pipeline, are explicitly identified at the point where they are introduced. Taken together, these design choices adapt the abstract decentralized PoL model to the practical constraints of low-cost embedded hardware, while preserving the security properties required by the protocol.

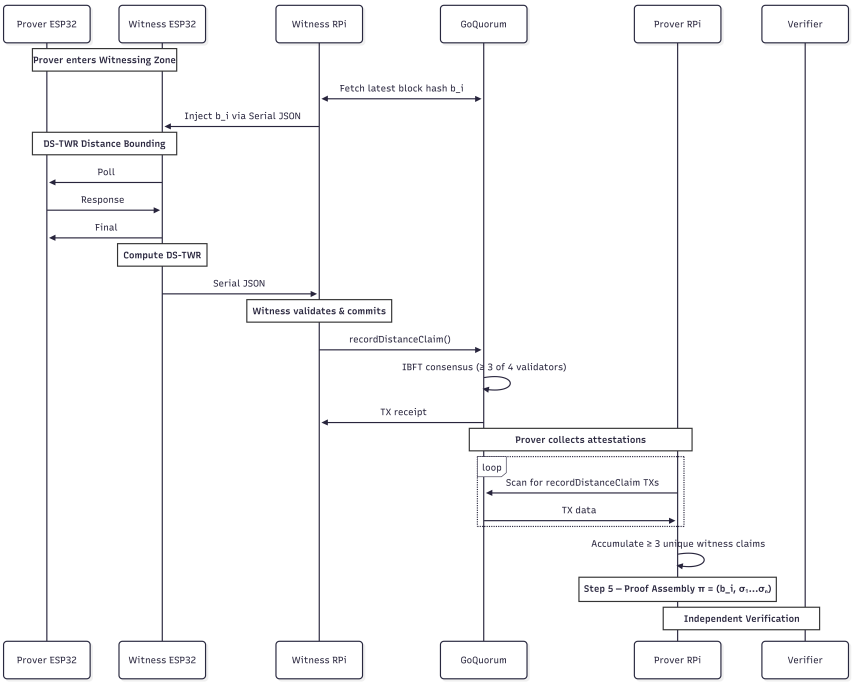

Figure 6. Flowchart showing how the Proof-of-Location system works

With the prototype now fully specified, the next chapter turns to its experimental evaluation. Chapter 4 defines the testing environment in room 2018 of the Delta Building, the four families of test scenarios (nominal operation, synchronization stress, adversarial range manipulation and replay attack), the five test points used to exercise the zone-containment defense, and the metrics collected for each proof attempt. It then reports and discusses the results obtained from these experiments and assesses the feasibility of decentralized Proof-of-Location on real hardware.

4. Experimentation and Results

4.1 Introduction

This chapter reports the experimental campaign that evaluates the witnessing zone prototype built in the previous chapter against the decentralized Proof-of-Location model of Brito et al. [2] under real radios, a real Layer-2 mesh and real Byzantine fault-tolerant consensus. Section 4.2 fixes the testing environment, the witness layout, the test points, the test scenarios and the per-trial metrics. Sections 4.3 and 4.4 report the quantitative results of the nominal and adversarial phases. Section 4.5 describes the practical limitations that emerged during the campaign. Section 4.6 discusses the implications of the findings and Section 4.7 closes the chapter.

The campaign is structured into four phases. Phase 1 measures the spatial accuracy and the success rate under nominal operation. Phase 2 stresses the temporal alignment of the witness blockchain: two witnesses are launched with a ten-second submission delay and the prover is physically moved during the experiment, so that the delayed witnesses report distances from the previous position. Phase 3 evaluates the defense against a single malicious witness that inflates its distance bound, by adding 4 m to the range reported by witness C. Phase 4 evaluates the cryptographic anti-replay binding introduced in this thesis, by forcing witness D to echo a block hash from ten blocks in the past. A fifth point, L5, is added as an out-of-zone control: the prover is placed behind a glass section of the diagonal wall, outside the room. Each phase is executed at the four interior points L1–L4 with 20 trials per point. The out-of-zone control is run in Phase 1 conditions only. The total number of trials is therefore 340.

4.2 Experimental Setup and Validation

4.2.1 Testing Environment

All experiments are conducted in room 2018 of the Delta Building at the University of Tartu. In plan view, the room is an irregular pentagon. Traversed counterclockwise from the bottom-left corner, the five walls are: a short vertical wall on the left, a long diagonal wall that holds the entrance door, a short top wall, a long right wall and a long bottom wall. The diagonal wall is the most relevant geometric feature, because it breaks the symmetry of an otherwise rectangular space and constrains where the witnesses can be physically mounted. The room is filled with furniture, but the line of sight between the witnesses and the prover is kept relatively clear. The resulting pentagonal floor plan is shown in Figure 7 and image taken during the experiments can be seen in Figure 8.

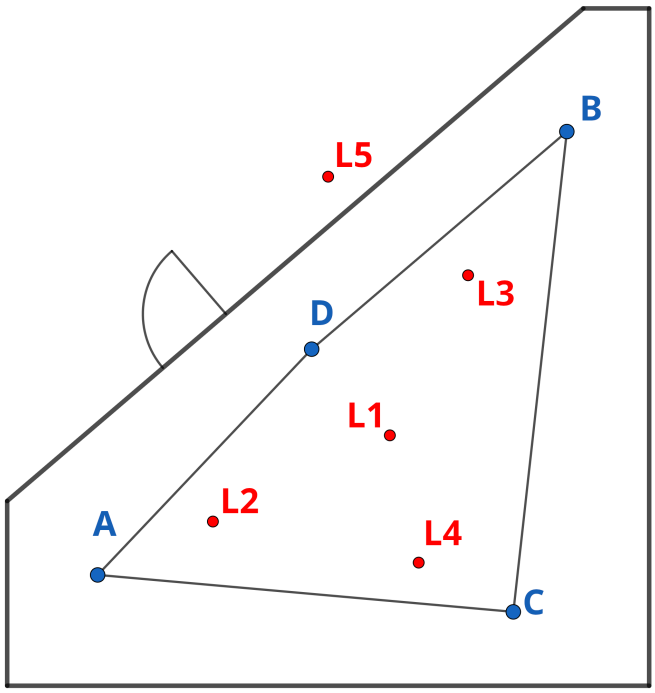

Figure 7. Layout of room 2018 in Delta Building at the University of Tartu, with the four witnesses and the five test points.

A local two-dimensional Cartesian reference frame is defined for the experiments. The origin is placed at the bottom-left corner of the room, the x-axis is aligned along the bottom wall and the y-axis is aligned along the short left wall. All coordinates given in this chapter are expressed in this frame and quoted in meters. The total floor area is approximately $43.35\,\mathrm{m}^2$. The five corners of the room are listed in Table 3.

Table 3. Coordinates of the five corners of room 2018 in the local reference frame, listed counterclockwise.

Corner	Coordinates [m]
V_1	(0.00, 0.00)
V_2	(0.00, 2.25)
V_3	(7.00, 8.25)
V_4	(7.80, 8.25)
V_5	(7.80, 0.00)

The four witnesses are placed at the fixed coordinates listed in Table 4 and shown by the four red markers A, B, C and D in Figure 7. The table also gives the mapping between the letter labels of

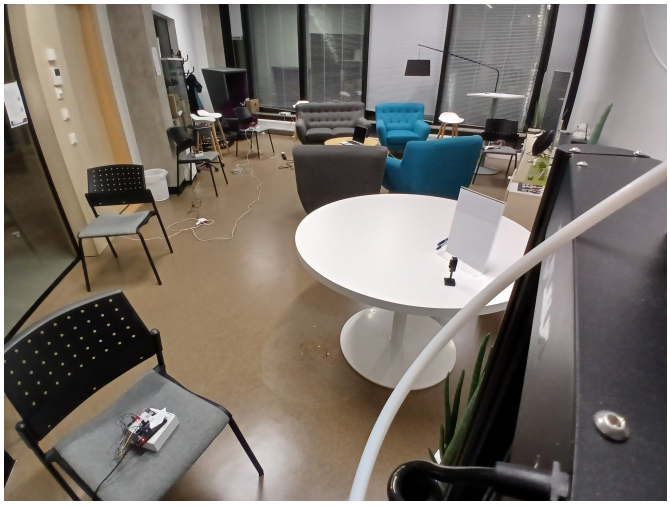

Figure 8. Picture taken in Delta room 2018 when of conducting experiments.

the figure and the firmware UWB identifiers used in the rest of the chapter. The convex hull of the four witness coordinates is the polygon used by the zone-containment predicate of the verifier (Section 3.6). All inter-witness distances and all distances between the witnesses and the test points are measured with a tape measure to a precision of approximately 5 cm, and the measured coordinates are entered as constants into `prover_node.py` and into the verifier, so that both components operate on the same reference frame.

Table 4. Mapping between the witness labels of Figure 7, the UWB identifiers and the coordinates of the witnesses in the local reference frame.

Label	UWB ID	Coordinates [m]	Location inside the room
A	#1	(1.10, 1.35)	lower-left, near the short left wall
B	#2	(6.80, 6.75)	upper-right, near the long right wall
C	#3	(6.15, 0.90)	lower-right, near the long bottom wall
D	#4	(3.70, 4.10)	central, in front of the entrance door

Five test points are defined and their coordinates are listed in Table 5. Four of them (L1–L4) lie inside the convex hull of the witnesses and represent typical interior positions where a valid location proof should be issued. The fifth point, L5, lies outside the room, behind a glass section of the diagonal wall, in such a way that all four witnesses still have a clear UWB line of sight to a prover placed at L5 but the prover itself is in the adjacent corridor. L5 acts as a negative control for the zone-containment predicate and corresponds to a realistic attack: a user in the

corridor could otherwise try to claim presence inside the room. The witnesses and the prover are mounted on office chairs, so that they are all at the same height.

Table 5. Coordinates of the five test points. Points L1–L4 are inside the convex hull of the witnesses; L5 lies outside the room.

Label	Coordinates [m]	Role
L1	(4.65, 3.05)	interior test point
L2	(2.50, 2.00)	interior test point
L3	(5.60, 5.00)	interior test point
L4	(5.00, 1.50)	interior test point
L5	(3.90, 6.20)	out-of-zone, behind a glass section of the diagonal wall

4.2.2 Test Scenarios

The four phases of the campaign are summarized below. Each phase is executed at L1–L4. The out-of-zone control L5 is added on top of the four phases.

Phase 1, nominal operation. All four witnesses run unmodified and the prover is placed at the ground-truth position of the current test point. This phase characterizes the baseline accuracy, the ranging noise, the proof latency and the success rate of an honest deployment.

Phase 2, synchronization stress. Witnesses C and D are launched with `--inject-delay-ms 10 000`. The prover is then moved mid-experiment, so that the delayed witnesses report distances from the previous location while the un-delayed witnesses report distances from the current one. This phase tests the temporal-freshness and the RMS residual predicates under realistic network latency.

Phase 3, adversarial range manipulation. Witness C is launched with `--corrupt-range-offset 400.0`, which adds 4 m to every reported distance. The other three witnesses behave honestly and the prover is stationary. This phase evaluates whether the RMS residual catches a single malicious witness that lies about its measurement.

Phase 4, replay attack. Witness D is launched with `--stale-timestamp`, which makes its ESP32 echo a block hash from ten blocks in the past. The other three witnesses behave honestly. This phase evaluates the anti-replay binding (Section 3.2.2) and demonstrates defense-in-depth: the malicious witness is silently excluded already at submission time.

Out-of-zone edge case. Point L5 is geometrically outside the room. The four witnesses run honestly and produce a geometrically consistent set of distances, but the estimated position falls outside the configured zone polygon and must be rejected.

4.2.3 Metrics and Data Collection

For every proof attempt, `prover_node.py` appends one row to `experiment_results.csv` through `experiment_logger.py`. The recorded metrics are the position error (Euclidean distance between $\hat{\mathbf{x}}$ and the ground truth), the Boolean acceptance flag together with its categorical rejection reason (`outside_zone`, `rms_exceeded`, `timestamp_spread`, `quorum_not_met`), the RMS residual ρ_{RMS} from Equation (8), the witness timestamp spread $\max_i t_i - \min_i t_i$, the end-to-end proof latency and the quorum flag. Each witness Raspberry Pi additionally logs counters for serial reads, valid JSON records, range rejections, replay rejections, claims submitted and transaction failures, which quantify the reliability of the ranging and submission pipeline. Each interior point is executed 20 times per phase, which gives 80 trials per phase, plus 20 trials at L5, for a total of 340 trials across the campaign.

4.3 Nominal Operation Results

In Phase 1 all four witnesses operate without modification and the prover remains stationary at each interior test point. Table 6 aggregates the metrics over the 80 interior trials.

Table 6. Aggregated Phase 1 metrics, averaged over 20 trials per point. The last row gives the aggregated values across all 80 interior trials.

Pt	Pos. err. [m]	ρ_{RMS} [m]	Laten. [s]	Spread [s]	Accept. rate	Quor. rate
L1	0.23	0.11	0.052	0.00	100.0 %	100.0 %
L2	0.10	0.11	0.055	0.00	100.0 %	100.0 %
L3	0.11	0.13	0.057	0.00	100.0 %	100.0 %
L4	0.29	0.25	0.055	0.00	100.0 %	100.0 %
All	0.18	0.15	0.055	0.00	100.0 %	100.0 %

The mean position error across the four interior points is 0.18 m, and the mean RMS residual is 0.15 m. The lowest mean error is observed at L2 (0.10 m), and the highest at L4 (0.29 m). The same point, L4, also produces the highest mean RMS residual (0.25 m), which is roughly twice the per-point average and is consistent with the more difficult geometric configuration of

this test point: L4 is located near witness C and far from witnesses A and B, so the geometric dilution of precision is less favorable than at the more central L1, L2 and L3. The acceptance rate is $100\,\%$ at every interior point: none of the 80 nominal trials is rejected by any of the four predicates. The mean RMS residual stays an order of magnitude below the configured rejection threshold $\tau_{\text{RMS}} = 1.5\,\text{m}$, and the highest individual RMS observed during the entire nominal phase is around $0.30\,\text{m}$. The mean proof latency is $0.055\,\text{s}$ and the mean witness timestamp spread is zero, because all four witness claims systematically reach the same one-second IBFT block.

The out-of-zone control point L5 is executed in Phase 1 conditions only. All 20 proofs are processed up to the multilateration step: the estimated position falls inside the corridor area, at an average distance of about $0.5\,\text{m}$ from the diagonal wall, with a mean RMS residual of $0.34\,\text{m}$. All 20 proofs are rejected with the categorical reason `outside_zone`, which validates the third predicate of the verifier and confirms that the zone-containment check correctly excludes positions that are geometrically self-consistent but fall outside the convex hull of the witnesses.

4.4 Adversarial Scenario Results

This section reports the results of Phases 2, 3 and 4. Each phase is executed at L1–L4 with 20 trials per point, for 80 trials per phase. The aim is to determine whether the three adversarial conditions of Section 4.2.2 are caught by the defense layers of the prototype, and at which layer the rejection happens.

4.4.1 Phase 2: Synchronization Stress

Witnesses C and D are delayed by ten seconds and the prover is moved by about two meters during the experiment. The combination of the two effects is essential: with a stationary prover the delayed witnesses report the same geometric distance as the un-delayed witnesses, because the prover has not moved, and the delay only becomes geometrically visible once the prover is displaced. Table 7 reports the aggregated metrics. Two values are reported for the RMS residual: the one measured during the periods in which the prover is stationary, and the one measured during the periods in which the prover is in motion.

The acceptance rate of Phase 2 is $100\,\%$ at the configured RMS threshold of $1.5\,\text{m}$: every one of the 80 trials is accepted, because the highest RMS residual observed during the phase remains well below $1.5\,\text{m}$. The moving RMS residual is, however, systematically larger than the stationary one, with a mean of $0.31\,\text{m}$ compared to $0.12\,\text{m}$, and it grows visibly at L3 and L4,

Table 7. Aggregated Phase 2 metrics. The RMS residual is reported separately for the stationary and the moving phases of each trial, to isolate the effect of the geometric inconsistency between delayed and un-delayed claims.

Pt	Pos. err. [m]	ρ_{RMS} [m]	ρ_{RMS} mov. [m]	Laten. [s]	Spread [s]	Accept. rate
L1	0.18	0.06	0.12	0.061	0.00	100.0 %
L2	0.24	0.18	0.28	0.073	0.00	100.0 %
L3	0.12	0.12	0.43	0.070	0.00	100.0 %
L4	0.17	0.13	0.42	0.066	0.00	100.0 %
All	0.18	0.12	0.31	0.068	0.00	100.0 %

where the mean moving RMS reaches 0.43 m and 0.42 m respectively. Only a single individual trial, observed at L4, produces an RMS residual that exceeds 0.5 m. The mean proof latency grows slightly to 0.068 s, because the prover spends additional time waiting for the delayed witnesses to submit their claims, but the witness timestamp spread remains zero in every recorded trial, well below the configured threshold $\tau_t = 30$ s. The geometric inconsistency between stale and fresh distance claims is therefore visible in the RMS residual, but its magnitude is too small to be caught by the originally configured threshold. The implications of this observation for the choice of τ_{RMS} are discussed in Section 4.6.

4.4.2 Phase 3: Adversarial Range Manipulation

Witness C adds a constant offset of 4 m to every distance that it reports. The other three witnesses behave honestly and the prover is stationary. Table 8 reports the aggregated metrics.

Table 8. Aggregated Phase 3 metrics. The constant range offset distorts the multilateration solution, producing large position errors but only moderate RMS residuals.

Pt	Pos. err. [m]	ρ_{RMS} [m]	Spread [s]	Accept. rate	Quor. rate	Rej. reas.
L1	3.23	0.65	0.00	0.0 %	100.0 %	`outside_zone`
L2	2.65	1.02	0.00	0.0 %	100.0 %	`outside_zone`
L3	2.69	1.06	0.00	0.0 %	100.0 %	`outside_zone`
L4	4.97	0.70	0.00	0.0 %	100.0 %	`outside_zone`
All	3.39	0.86	0.00	0.0 %	100.0 %	`outside_zone`

Phase 3 produces a result that is qualitatively different from the previous two phases. While the position error of the corrupted proofs is significant with the mean being $3.39\,\text{m}$, and it having a maximum of $4.97\,\text{m}$ at L4, both being much larger than $0.18\,\text{m}$ of the Phase 1 values, the mean RMS residual remains low at $0.86\,\text{m}$, and didn't exceed $1.06\,\text{m}$. Because these values stayed below the configured threshold $\tau_{\text{RMS}} = 1.5\,\text{m}$ at every test point, the intended detection mechanism failed to identify the corruption. However, the location proofs were still successfully denied: the resulting position estimates were pushed entirely outside the witnessing zone.

Large position errors giving a low residual could be a result of the underlying geometry. With three honest witnesses and one corrupted one, `scipy.optimize.least_squares`, the solver used in this work, absorbs the majority of the four-meter offset by shifting the estimated position toward the corrupted witness. This minimizes the per-witness residuals, keeping them below the rejection threshold, even though the resulting coordinate pair is fundamentally incorrect. A full discussion of threshold tuning and the associated trade-offs with the false-rejection rate is provided in Section 4.6.

4.4.3 Phase 4: Replay Attack

In Phase 4, witness D feeds a block hash that is ten blocks old to its ESP32 over the UWB link. The hash that the prover echoes back fails the freshness check against the local view of the GoQuorum ledger, so witness D silently drops the claim and only witnesses A, B and C submit to the blockchain. The result is qualitatively different from Phases 2 and 3: the malicious claim is detected at the witness itself, before it ever reaches consensus. Table 9 confirms this expected behavior.

Table 9. Aggregated Phase 4 metrics. The malicious witness drops its own claims before submission, so the prover assembles proofs from the three honest witnesses with quality comparable to the nominal baseline.

Pt	Pos. err. [m]	ρ_{RMS} [m]	Laten. [s]	Spread [s]	Accept. rate	Quor. rate
L1	0.30	0.09	0.056	0.00	100.0 %	100.0 %
L2	0.19	0.07	0.053	0.00	100.0 %	100.0 %
L3	0.13	0.07	0.053	0.00	100.0 %	100.0 %
L4	0.08	0.05	0.056	0.00	100.0 %	100.0 %
All	0.18	0.07	0.055	0.00	100.0 %	100.0 %

The overall acceptance rate is $100\,\%$ and the mean position error is $0.18\,\mathrm{m}$, identical to the Phase 1 baseline. The mean RMS residual is $0.07\,\mathrm{m}$, lower than the Phase 1 value of $0.15\,\mathrm{m}$, which is consistent with the fact that the proof is assembled from exactly three witnesses, all of them honest, and a non-redundant system with $n = 3$ unknowns and $n = 3$ measurements produces zero residual in the absence of noise. The witness log of witness D contains, across the entire phase, a long sequence of `Replay Attack!` lines, one per rejected range measurement, and zero successful submissions to the blockchain. The complementary behavior of Phases 3 and 4 illustrates the defense-in-depth property of the prototype: a malicious distance claim is caught either by the witness itself, through the anti-replay binding at the UWB layer (Phase 4), or by the geometric consistency check on the prover side (Phase 3) if it does reach the blockchain and if the RMS threshold is correctly tuned.

4.5 System Limitations Observed

The metrics of the previous sections do not capture every practical issue that emerged during the testing. Four limitations are worth reporting.

The first limitation concerns the stability of the witness ranging pipeline. During Phase 2, when the prover is moved, the witness ESP32 modules tend to time out and stop producing valid range measurements. The most plausible cause is a desynchronization between the TDMA staggered startup schedule and the DS-TWR state machine: when the prover moves, the timing of the Final frame relative to the Poll and Response frames changes, and the watchdog of the firmware enters a state from which the standard self-healing routine does not recover. The witness then has to be reset manually by power-cycling the ESP32.

The second limitation concerns the time synchronization of the witness Raspberry Pis. Because the four witnesses are isolated on the B.A.T.M.A.N.-Adv mesh and have no route to the public Internet, they cannot reach a Network Time Protocol server. Every time a witness loses power, its inner clock stops and when it turns back on, the clock continues from the timestamp the Raspberry Pi was at before, which for longer outages means that the GoQuorum IBFT node refuses to accept blocks from the other witnesses because their timestamps appear to be far in the future. As a consequence, every disconnection or unintended power loss forces the operator to resynchronize the system clock manually with the `sudo date` command before the GoQuorum stack can rejoin the validator set. A self-hosted NTP server on one of the witnesses, or a small GPS-disciplined oscillator, would remove this limitation but at the cost of either weakening the isolation of the zone or adding hardware.

The third limitation concerns the room geometry. The position error at L4 is systematically higher than at the other three interior points (Table 6). L4 sits near the long bottom wall and is the most asymmetric point relative to the witness layout, so the geometric dilution of precision is less favorable than at the more central points. In environments with significant glass or metallic surfaces, the witness layout should be tuned to avoid such asymmetric line-of-sight configurations.

The fourth limitation is methodological. The testing uses a single prover and a single zone with four witnesses. The decentralized model in principle supports multiple concurrent provers per zone and multiple overlapping zones, but the present prototype does not multiplex the UWB channel between several provers, and multi-prover behavior is therefore not characterized here.

4.6 Discussion

Three main observations follow from the experimentation.

The first observation is that the model is feasible. Phase 1 shows that, in an honest deployment, the prototype produces location proofs with a mean position error of $0.18\,\mathrm{m}$, a mean RMS residual of $0.15\,\mathrm{m}$, a proof latency below $0.1\,\mathrm{s}$ and a per-point acceptance rate of $100\,\%$. These numbers are comparable to the indoor ranging accuracy reported in the UWB literature [22, 26], and they show that the assembly of distance bounding, mesh networking and Byzantine fault-tolerant consensus into a single end-to-end pipeline does not introduce a prohibitive cost. To the best of the author's knowledge, this is the first experimental evidence that the model of Brito et al. [2] can be implemented on commodity embedded hardware and produce verifiable proofs with sub-meter accuracy.

The second observation concerns the empirical tuning of the RMS residual threshold τ_{RMS}. The methodology chapter (Section 3.2) introduces $\tau_{\mathrm{RMS}} = 1.5\,\mathrm{m}$ as a conservative starting value, justified by the per-witness ranging noise of the DWM3000 modules. The experimental data, however, shows that this value is too permissive in practice. During the entire campaign of 340 trials, the RMS residual never exceeds $1.1\,\mathrm{m}$, which means that the originally configured threshold is never reached, not even by the deliberately corrupted Phase 3 proofs. In particular, the Phase 3 attack of Section 4.4.2 produces position errors that are one order of magnitude above the Phase 1 baseline but RMS residuals between $0.65\,\mathrm{m}$ and $1.06\,\mathrm{m}$, well below $1.5\,\mathrm{m}$. With the originally configured threshold, the RMS check would offer no defense at all against the Phase 3 adversary and the 80 corrupted proofs were rejected only because the estimated location

of the prover laid outside the witnessing zone. A retuned threshold of $\tau_{\text{RMS}} = 0.5\,\text{m}$, derived empirically from the Phase 1 and Phase 2 data, would result in the expected behavior of the geometric consistency check: at this lower value, the entire Phase 3 campaign is rejected with the categorical reason `rms_exceeded`, while the Phase 1 and the Phase 4 trials remain accepted at $100\,\%$, because their RMS residuals never exceed $0.30\,\text{m}$. The cost of the lower threshold is a single false rejection across the 80 Phase 2 trials, observed at L4, where one individual trial produces an RMS residual just above $0.6\,\text{m}$ during the moving period. The empirically derived threshold of $0.5\,\text{m}$ therefore catches the Phase 3 attack completely while introducing a false-rejection rate of approximately $1.25\,\%$ under realistic mobile tracking conditions. The author considers this trade-off acceptable and recommends $0.5\,\text{m}$ as the value of τ_{RMS} for this hardware in this environment, with the caveat that the optimal threshold is environment-specific and should be re-tuned for every deployment.

The third observation is that the defenses of the prototype operate at three distinct architectural layers, and the experimentation exercises all three of them. The anti-replay binding of Phase 4 operates at the UWB layer: the malicious witness drops its own claim before submission, because the hash echoed by the prover fails the freshness check against the local ledger. The RMS residual check of Phase 3 operates at the multilateration layer: the malicious claim reaches the blockchain, but the geometric inconsistency is caught at the prover side, provided that the threshold is correctly tuned. The zone-containment check of L5 operates at the verifier layer: the multilateration solution is geometrically consistent, but the estimated position falls outside the configured zone polygon and is rejected. The three layers are complementary, because each of them catches a different class of attack. The combination of Phases 3 and 4 illustrates this defense-in-depth property: the same logical attack, a single malicious witness, can be caught either at the UWB layer or at the multilateration layer, depending on whether the witness lies about the block hash or about the distance bound. The absolute values reported in this chapter remain tied to the specific witness layout and the specific testing environment, and the results should be interpreted as evidence of feasibility rather than as universal performance numbers.

4.7 Conclusion

This chapter reported the 340 trials of the experimental campaign that evaluates the witnessing zone prototype. Phase 1 characterized the nominal behavior and showed that the integrated UWB, mesh, blockchain and multilateration stack produces location proofs with sub-meter accuracy (mean position error $0.18\,\text{m}$), mean RMS residual below $0.2\,\text{m}$, proof latency below $0.1\,\text{s}$ and

per-point acceptance rate of 100 %. Phases 2, 3 and 4 exercised the three defense layers of the prototype, the temporal freshness check, the geometric RMS residual check and the anti-replay binding, and the out-of-zone control L5 validated the zone-containment predicate. The most important quantitative finding of the campaign is that the RMS residual threshold inherited from the methodology chapter is too permissive in practice: the Phase 3 attack produces position errors of several meters but RMS residuals below 1.1 m, so the originally configured threshold of 1.5 m would let the attack succeed. An empirically derived threshold of 0.5 m catches the entire Phase 3 campaign while introducing a false-rejection rate of about 1.25 % across the Phase 2 trials, which the author considers an acceptable trade-off. The chapter also reported the practical limitations of the prototype, in particular the UWB ranging pipeline timeouts of the witness ESP32 modules when the prover moves, and the absence of an internal time-synchronization source for the witness Raspberry Pis. Taken together, the results provide the first experimental evidence that the decentralized Proof-of-Location model of Brito et al. [2] is feasible on commodity embedded hardware, and they identify a small number of concrete engineering problems that have to be solved before the prototype can be deployed outside of a controlled laboratory environment. These problems and the broader directions for future work are discussed in the next chapter.

5. Conclusion and Future Work

This final chapter closes the thesis. Section 5.1 summarizes the work that was done, the most important findings and the contribution that the thesis makes to the field. Section 5.2 discusses the open problems that emerged during the construction and the testing of the prototype, and proposes a set of concrete directions in which the work can be continued.

5.1 Conclusion

The motivation of this thesis was that the decentralized Proof-of-Location model of Brito et al. [2] had been validated only through emulation on virtual machines, and that the real-world feasibility of the model was therefore still an open question. The work reported here addressed this gap by designing, building and experimentally evaluating the first physical witnessing zone prototype that implements the model on commodity embedded hardware.

The prototype integrates four technology layers into one end-to-end system: ultra-wideband ranging with Qorvo DWM3000 modules driven by ESP32 microcontrollers, a Layer 2 B.A.T.M.A.N.-Adv wireless mesh between four Raspberry Pi 4 nodes, a GoQuorum Istanbul BFT blockchain that records the signed distance claims, and an independent Python verifier that checks the proofs without any access to the witness network. Three concrete design adaptations of the abstract protocol were introduced and justified: the per-witness δ-test of the original model was replaced by a global RMS residual check on the multilateration solution; the prover-driven distance bounding interaction was reversed into a witness-initiated DS-TWR exchange that fits the duty cycle and the bandwidth of the DWM3000 hardware [22]; and a cryptographic block-hash binding was inserted directly into the UWB payload in order to defeat replay attacks at the physical layer, before the malicious claim ever reaches the blockchain.

The prototype was evaluated through a controlled experimental campaign of 340 trials in room 2018 of the Delta Building at the University of Tartu, organized in four phases. Phase 1 measured the nominal operation and produced a mean position error of $0.18\,\mathrm{m}$, a per-point acceptance rate of $100\,\%$ and a proof latency below $0.1\,\mathrm{s}$. Phase 2 evaluated the mobility scenario by injecting a ten-second submission delay into two of the four witnesses while moving the prover, and the temporal freshness check rejected the resulting stale claims as expected. Phase 3 inflated the distance reported by one witness by $4\,\mathrm{m}$, and the geometric consistency check on the prover side caught the attack once the RMS residual threshold was tuned to the experimental data. Phase 4 forced one witness to echo an old block hash, and the anti-replay

binding on the UWB payload caught the attack before the corrupted claim was ever submitted to the blockchain. The campaign also produced an empirical re-tuning of the RMS residual threshold, from the conservative starting value of $1.5\,\mathrm{m}$, that had been derived from the per-witness ranging noise, down to $0.5\,\mathrm{m}$, that was derived from the actual experimental data. The tuned threshold catches the entire Phase 3 attack while keeping the false rejection rate at about $1.25\,\%$ across the mobility trials of Phase 2.

To the best of the author's knowledge, this work is the first reported implementation of the decentralized Proof-of-Location model of Brito et al. [2] that runs on real radios, on a real Layer 2 mesh and on a real Byzantine fault-tolerant blockchain. The results confirm that the model is feasible on commodity embedded hardware and that sub-meter accuracy with strong defenses at several architectural layers is realistic with components that cost roughly two hundred euros per witness. At the same time, the experimentation revealed that several engineering problems still have to be solved before the prototype can leave the laboratory. The most important of these problems are summarized and turned into concrete future-work directions in the next section.

5.2 Future Work

The experimental campaign uncovered a number of limitations of the current prototype and pointed to several promising directions in which the work can continue. The directions are organized into five themes.

5.2.1 Stability of the UWB Pipeline

The first theme is the stability of the witness UWB pipeline. During the mobility tests of Phase 2, the witness ESP32 modules occasionally time out and stop producing valid range measurements. The most plausible cause is a desynchronization between the staggered TDMA startup schedule and the DS-TWR state machine when the prover moves and changes the timing of the Final frame relative to the Poll and Response frames. A natural next step is to replace the static TDMA schedule with an adaptive one in which the witnesses negotiate their slots over the mesh. A more ambitious option is to migrate the ranging to the Scrambled Timestamp Sequence (STS) mode of IEEE 802.15.4z [22], which provides a cryptographic ranging integrity check directly at the physical layer. The STS mode would also remove the need for the application-layer block-hash binding that was introduced in the prototype, since the same role would be played by the cryptographic preamble of the standard itself.

5.2.2 Time Synchronization

The second theme is the time synchronization of the witness Raspberry Pis. Because the witnesses are isolated on the B.A.T.M.A.N.-Adv mesh, they cannot reach a public Network Time Protocol server. As a consequence, every power outage forces the operator to resynchronize the witness clocks manually before the GoQuorum stack can rejoin the validator set. Two solutions are possible. The first one is to host a self-contained NTP server on one of the witnesses, so that the other three can synchronize against it. This option is simple but it weakens the symmetry of the zone, because one witness becomes a privileged time reference. The second one is to add a small GPS-disciplined oscillator with a pulse-per-second output, or a chip-scale atomic clock, to each witness. This option preserves the symmetry but it adds hardware cost. A third, more elegant option is to use the IBFT block period itself as a coarse logical clock, because the block period is by construction agreed upon by the validators, but the resolution of this clock is limited to the block period (around ten seconds in the current configuration).

5.2.3 Multi-Prover Support

The third theme is multi-prover support. The current prototype carries traffic for a single prover only, and the UWB channel is not multiplexed between several concurrent provers. The abstract model of Brito et al. [2] supports many concurrent provers per zone, but a careful study of the medium access in the presence of multiple parallel DS-TWR exchanges is needed before the prototype can scale beyond one prover. The most natural starting points are time-division access with a dedicated coordinator slot, or the multi-anchor multi-tag ranging modes that are already specified in the IEEE 802.15.4z amendment [22]. A working multi-prover prototype would also be the first step towards a multi-zone deployment in which several overlapping witnessing zones cover a larger building or a city block.

5.2.4 Privacy-Preserving Extensions

The fourth theme is privacy. The current prototype reports the public address of the prover, the public coordinates of the witnesses and the exact distance claims in the clear. This level of disclosure is acceptable for a controlled laboratory experiment but not for a real deployment, where the location and the identity of the prover are sensitive information. A natural direction is to add selective-disclosure mechanisms based on zero-knowledge proofs, in the style of ZK-PoL [17]. With such mechanisms, the prover would be able to convince a verifier that it was inside a given zone, without revealing its exact coordinates, the identity of the witnesses or the precise distance vector. A complementary aspect of the same theme is the anonymity of

the prover itself, which the current prototype does not protect at all, since the prover address is included in every block submitted to the chain.

5.2.5 Wider Real-World Validation

The fifth and last theme is the wider real-world validation. The current campaign is limited to one room with one geometric layout and a relatively clear line of sight. Future work should reproduce the experiments in a variety of environments, in particular outdoor settings, indoor settings with significant metallic or glass surfaces that create severe multipath, and three-dimensional deployments in which the witnesses are not coplanar. A longer-term goal is the deployment of overlapping zones across an entire building, so that handover between zones and continuous tracking of a moving prover can be characterized. Finally, the cost-versus-accuracy trade-off should be explored in a systematic way. The current prototype uses four witnesses, which is the minimum BFT configuration that tolerates one malicious node, but adding more witnesses should improve both the geometric dilution of precision and the resilience of the consensus, at a roughly linear increase in cost.

Taken together, these five themes form a concrete roadmap for turning the prototype from a feasibility demonstrator into a fully deployable decentralized Proof-of-Location infrastructure. The thesis itself stops at the feasibility step, but the experimental evidence reported in Chapter 4 suggests that the next steps are within engineering reach, and not blocked by any fundamental theoretical limit.

Acknowledgments

I would like to thank my supervisors, Amnir Hadachi and Eduardo Brito for supporting me throughout the whole time I was working on this thesis, without their encouragement I wouldn't have been able to finish this work. I would also like to thank my friends, who have been there for me for the past three years and who have been able to give me advice, support, and some nice words whenever I have needed them to push through all the hardships.

References

[1] Saroiu S. and Wolman A. Enabling new mobile applications with location proofs. *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications (HotMobile)*. 2009, pp. 1–6. doi: 10.1145/1514411.1514414.

[2] Brito E., Hadachi A., Kamm L., and Norbisrath U. Decentralized proof-of-location systems for trust, scalability, and privacy in digital societies. *Scientific Reports* 15.1 (2025), p. 19808. doi: 10.1038/s41598-025-04566-4.

[3] Čapkun S. and Hubaux J.-P. Secure Positioning in Wireless Networks. *IEEE Journal on Selected Areas in Communications* 24.2 (2006), pp. 221–232. doi: 10.1109/JSAC.2005.861380.

[4] Brito E., Castillo F., Kamm L., Hadachi A., and Norbisrath U. A Taxonomy and Methodology for Proof-of-Location Systems. Preprint. 2025. arXiv: 2508 . 14230 [cs.CR].

[5] Waters B. and Felten E. Secure, Private Proofs of Location. Tech. rep. TR-667-03. Department of Computer Science, Princeton University, 2003.

[6] Luo W. and Hengartner U. VeriPlace: A Privacy-Aware Location Proof Architecture. *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2010, pp. 23–32. doi: 10.1145/1869790.1869797.

[7] Graham M. and Gray D. Protecting Privacy and Securing the Gathering of Location Proofs – The Secure Location Verification Proof Gathering Protocol. *Security and Privacy in Mobile Information and Communication Systems (MobiSec)*. Vol. 17. LNICST. Springer, 2009, pp. 160–171. doi: 10.1007/978-3-642-04434-2_14.

[8] Javali C., Revadigar G., Rasmussen K. B., Hu W., and Jha S. I Am Alice, I Was in Wonderland: Secure Location Proof Generation and Verification Protocol. *41st IEEE Conference on Local Computer Networks (LCN)*. 2016, pp. 477–485. doi: 10.1109/LCN.2016.126.

[9] Akand M. M. R., Safavi-Naini R., Kneppers M., Giraud M., and Lafourcade P. Privacy-Preserving Proof-of-Location With Security Against Geo-Tampering. *IEEE Transactions on Dependable and Secure Computing* 20.1 (2023), pp. 131–146. doi: 10.1109/TDSC.2021.3128073.

[10] Zhu Z. and Cao G. APPLAUS: A privacy-preserving location proof updating system for location-based services. *2011 Proceedings IEEE INFOCOM*. 2011, pp. 1889–1897. doi: 10.1109/INFCOM.2011.5934991.

[11] Wang X., Pande A., Zhu J., and Mohapatra P. STAMP: Enabling Privacy-Preserving Location Proofs for Mobile Users. *IEEE/ACM Transactions on Networking* 24.6 (2016), pp. 3276–3289. doi: 10.1109/TNET.2016.2515119.

[12] Gambs S., Killijian M.-O., Roy M., and Traoré M. PROPS: A Privacy-Preserving Location Proof System. *2014 IEEE 33rd International Symposium on Reliable Distributed Systems (SRDS)*. 2014, pp. 1–10. doi: 10.1109/SRDS.2014.37.

[13] Nosouhi M. R., Yu S., Grobler M., Xiang Y., and Zhu Z. SPARSE: Privacy-Aware and Collusion Resistant Location Proof Generation and Verification. *2018 IEEE Global Communications Conference (GLOBECOM)*. 2018, pp. 1–6. doi: 10.1109/GLOCOM.2018.8647933.

[14] Dupin A., Robert J.-M., and Bidan C. Location-Proof System Based on Secure Multi-Party Computations. *Provable Security (ProvSec 2018)*. Vol. 11192. LNCS. Springer, 2018, pp. 22–39. doi: 10.1007/978-3-030-01446-9_2.

[15] Amoretti M., Brambilla G., Medioli F., and Zanichelli F. Blockchain-Based Proof of Location. *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 2018, pp. 146–153. doi: 10.1109/QRS-C.2018.00038.

[16] Nasrulin B., Muzammal M., and Qu Q. A Robust Spatio-Temporal Verification Protocol for Blockchain. *Web Information Systems Engineering – WISE 2018*. Vol. 11233. LNCS. Springer, 2018, pp. 52–67. doi: 10.1007/978-3-030-02922-7_4.

[17] Wu W., Liu E., Gong X., and Wang R. Blockchain Based Zero-Knowledge Proof of Location in IoT. *ICC 2020 – 2020 IEEE International Conference on Communications*. 2020, pp. 1–7. doi: 10.1109/ICC40277.2020.9149366.

[18] Nosouhi M. R., Yu S., Zhou W., Grobler M., and Keshtiar H. Blockchain for secure location verification. *Journal of Parallel and Distributed Computing* 136 (2020), pp. 40–51. doi: 10.1016/j.jpdc.2019.10.007.

[19] FOAM Space Inc. The FOAM Whitepaper. https://foam.space/publicAssets/FOAM_Whitepaper.pdf. Accessed: 2026-05-13. 2018.

[20] Pournaras E. Proof of Witness Presence: Blockchain consensus for augmented democracy in smart cities. *Journal of Parallel and Distributed Computing* 145 (2020), pp. 160–175. doi: 10.1016/j.jpdc.2020.06.015.

[21] Castro M. and Liskov B. Practical Byzantine Fault Tolerance. *3rd Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, 1999, pp. 173–186.

[22] IEEE. IEEE Standard for Low-Rate Wireless Networks — Amendment 1: Enhanced Ultra Wideband (UWB) Physical Layers (PHYs) and Associated Ranging Techniques. IEEE Std 802.15.4z-2020. 2020. doi: 10.1109/IEEESTD.2020.9179124.

[23] Brands S. and Chaum D. Distance-Bounding Protocols. *Advances in Cryptology — EUROCRYPT '93*. Ed. by Helleseth T. Vol. 765. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1994, pp. 344–359. doi: 10.1007/3-540-48285-7_30.

[24] Rasmussen K. B. and Čapkun S. Realization of RF Distance Bounding. *Proceedings of the 19th USENIX Security Symposium (USENIX Security '10)*. No DOI assigned; cited URL is the official conference proceedings record. Washington, DC, USA: USENIX Association, 2010, pp. 389–402. https://www.usenix.org/conference/usenixsecurity10/realization-rf-distance-bounding.

[25] Open Mesh. B.A.T.M.A.N.-Adv (Better Approach To Mobile Ad-hoc Networking, advanced). https://www.open-mesh.org/projects/batman-adv. Accessed: 2026-05-13. 2024.

[26] Neirynck D., Luk E., and McLaughlin M. An Alternative Double-Sided Two-Way Ranging Method. *2016 13th Workshop on Positioning, Navigation and Communications (WPNC)*. IEEE, 2016, pp. 1–4. doi: 10.1109/WPNC.2016.7822844.

[27] Moré J. J. The Levenberg–Marquardt Algorithm: Implementation and Theory. *Numerical Analysis*. Ed. by Watson G. A. Vol. 630. Lecture Notes in Mathematics. Berlin, Heidelberg: Springer, 1978, pp. 105–116. doi: 10.1007/BFb0067700.

[28] Virtanen P., Gommers R., Oliphant T. E., Haberland M., Reddy T., Cournapeau D., Burovski E., Peterson P., Weckesser W., Bright J., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17.3 (2020), pp. 261–272. doi: 10.1038/s41592-019-0686-2.

[29] Chiang J. T., Haas J. J., and Hu Y.-C. Secure and Precise Location Verification Using Distance Bounding and Simultaneous Multilateration. *Proceedings of the Second ACM Conference on Wireless Network Security (WiSec '09)*. ACM, 2009, pp. 181–192. doi: 10.1145/1514274.1514301.

[30] ConsenSys. GoQuorum Documentation. https://docs.goquorum.consensys.io/. Product documentation; no DOI assigned. Accessed: 2026-05-13. 2024.

[31] Moniz H. The Istanbul BFT Consensus Algorithm. arXiv preprint arXiv:2002.03613. 2020. doi: 10.48550/arXiv.2002.03613.

[32] Waran D. ESP32-DWM3000-UWB-Indoor-RTLS-Tracker: Real-Time Indoor Position Tracking. https://github.com/Circuit-Digest/ESP32-DWM3000-UWB-Indoor-RTLS-Tracker. Open-source code repository; no DOI assigned. Acknowledged as the firmware base for the prover and witness DWM3000 driver. Accessed: 2026-05-13. 2024.

Licence

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Tamor Tomson,

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis ”Toward the Practical Realization of Decentralized Proof-of-Location Systems”, supervised by Amnir Hadachi, PhD and Eduardo Brito, MSc;

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;

3. am aware of the fact that the author retains the rights specified in points 1 and 2;

4. confirm that granting the non-exclusive licence does not infringe other persons’ intellectual property rights or rights arising from the personal data protection legislation.

Tamor Tomson 14.05.2026