



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna Archivio istituzionale della ricerca

Proof of Location through a Blockchain Agnostic Smart Contract Language

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Bonini, M., Zichichi, M., Ferretti, S., D'Angelo, G. (2023). Proof of Location through a Blockchain Agnostic Smart Contract Language. 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA : IEEE COMPUTER SOC [10.1109/ICDCSW60045.2023.00016].

Availability:

This version is available at: <https://hdl.handle.net/11585/961828> since: 2024-02-26

Published:

DOI: <http://doi.org/10.1109/ICDCSW60045.2023.00016>

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

Proof of Location through a Blockchain Agnostic Smart Contract Language

Michele Bonini*, Mirko Zichichi[†], Stefano Ferretti[‡], Gabriele D'Angelo*

*Department of Computer Science and Engineering, University of Bologna, Italy

[†]Ontology Engineering Group, Universidad Politécnica de Madrid, Spain

[‡]Department of Pure and Applied Sciences, University of Urbino Carlo Bo, Italy

michele.bonini2@studio.unibo.it, mirko.zichichi@upm.es, stefano.ferretti@uniurb.it, g.dangelo@unibo.it

Abstract—Location-based services are at the heart of many applications that individuals use every day. However, there is often no guarantee of the truthfulness of users' location data, since this information can be easily spoofed without a proof mechanism. In distributed system applications, preventing users from submitting counterfeit locations becomes even more challenging because of the lack of a central authority that monitors data provenance. In this work, we propose a decentralized architecture based on blockchains and decentralized technologies, offering a transparent solution for Proof of Location (PoL). We specifically address two main challenges, i.e., the issuing process of the PoL and the proof verification. We describe a smart contract based implementation in Reach, a blockchain-agnostic smart contract language, and the tests we conducted on different blockchains, i.e. Ethereum, Polygon, and Algorand, measuring latency and costs due to the payment of fees. Results confirm the viability of the proposal.

Index Terms—Distributed Ledger Technology, Decentralized File Storage, Distributed Hash Table, Keyword Search, Smart Contracts

I. INTRODUCTION

Nowadays, many users' activities are supported by a different number of mobile applications, leveraging their position to offer specific location-based services. For example, trustworthy crowd-sourcing of urban or environmental obstacles for accessibility purposes [1], [2], customer-loyalty reward systems that offer discounts to users who frequently visit the shop, privacy-preserving contact tracing [3].

These kinds of systems have at least three issues to cope with. First, some level of trust is needed in the user that crowd-sources some data related to a certain position. This led to the idea of a *Proof-of-Location* (PoL) because the location could be easily spoofed [4]. Second, there is the need to ensure, on the other hand, some privacy guarantees to the users that generate data, so as to avoid everyone being entitled to know a specific user location at a certain time. Third, the usual approach is to resort to a centralized system, where a single entity is responsible for collecting and storing data, users that generated them, and their associated position representing, somehow, a PoL. While this solution can help in dealing with the two issues above, it raises some concerns on personal data

sovereignty, as location data is one of the most sensitive cases with respect to users' data exploitation [5], [6].

With this in view, in this paper we propose a decentralized Proof of Location (PoL) system, based on blockchain and distributed storage technologies. Our system is designed to be decentralized, so as to avoid the presence of single point of failures and to reduce its typical risks [7]. Arguably, the centralization of power in a few entities' hands is likely to significantly threaten many aspects of location-based services' users, e.g., information privacy, censorship. Through our solution, users can generate a Location-Proof (LP) (i.e. the certificate generated by the PoL service) in a private, decentralized, and distributed way, using nearby users as witnesses. In particular, the proposed system exploits (i) IPFS [8] to store data; (ii) a Distributed Hash Table (DHT) organized as a hypercube [2] to manage all the discovery activities to locate, witness, and verify that the LP is correct; (iii) smart contracts to automatically handle all the interactions among the involved entities in the LP generation and verification.

We tested our decentralized application on different smart-contract-enabled blockchains, since many exist and with very different characteristics. Specifically, we chose Ethereum and Algorand, also using a layer-2 solution proposed by Polygon chain. During the tests, we made a performance analysis to evaluate speed and cost transaction metrics. In order to develop the smart contracts source code only once, we used Reach, a blockchain-agnostic language that builds smart contracts for each considered blockchain. Our results demonstrate the viability of the proposal and suggest that good performance can be obtained through the use of Algorand and layer-2 Ethereum based solutions.

In summary, the contribution of this work concerns the creation of a PoL system which prevents fake-location submission and allows data storing in a decentralized architecture and data integrity verification through the use of a distributed ledger and smart contracts.

The remainder of this paper is organized as follows. Section II provides the necessary background and discusses some related works. Section III presents the system model. Section IV focuses on the smart-contracts we built to implement the system. Section V provides a discussion on the experimental evaluation we conducted to assess the system implementation. Finally, Section VI provides some concluding remarks.

This work received funding from the EU H2020 R&D programme under the MSCA ITN EJD grant agreement No 814177 Law Science and Technology Joint Doctorate - RIoE.

II. BACKGROUND AND RELATED WORKS

A. Data Storage and Discovery

The need for verifiability and untamperability, in our application scenario, suggests resorting to Distributed Ledger Technologies (DLTs) as a main underlying system. However, since storing data directly on a ledger can be expensive and time-consuming, some strategies have been proposed to store data inside Decentralized File Storage (DFS) systems, such as IPFS, and then add specific hash pointers into the DLT [9]. Interplanetary File System (IPFS) is a DFS system that allows P2P file sharing to every node that wants to participate. The IPFS protocol assigns each object to a unique address called Content Identifier (CID) built hashing the file content.

In order to ease the discovery of data, we included in the system a Hypercube DHT [2]. The Hypercube is a Distributed Hash Table (DHT) that, similarly to other content addressable networks, organizes peers (and contents) in a n -dimensional Cartesian coordinate system. Each node is responsible for a specific keyword set and the related content. Upon request for data lookup based on some specific keywords, a specific Peer-to-Peer (P2P) protocol allows identifying the node that stores information where such data is located. In our system, data are located in the DFS and the related CIDs, that have the twofold role of content identifier and digest to check the validity of the data, are stored in the DLT (for verifiability purposes) and in the Hypercube (for content retrieval purposes) [10]. More in detail, our system stores in the Hypercube the LP data that verifiers have validated, accessible through the CID, together with all additional application-dependent information. For instance, we developed a use-case application where users can report critical points in an urban area or environmental issues, such as illegally abandoned wastes.

B. Reach: a blockchain-agnostic language

Reach [11] is a high-level language released in 2020, similar to Javascript, which allows the creation of Decentralized Applications (DApps) on a specific blockchain chosen by the programmers. Reach is *blockchain agnostic*, since it is possible to run a DApp in different blockchains without any code change. One of the big advantages of Reach is the verification process of the written code in order to guarantee a safe and efficient program, e.g., the “token linearity property” verification which requires an empty balance when the smart contract terminates. When a DApp is built, the Reach builder produces the smart contract and an intermediate component, that we refer to as middleware. This middleware offers a frontend system and a backend that manages the connection between the middleware itself and the smart contract. All the complexity is hidden, and for this reason, the developers do not have to specify the details about how a smart contract works at a low level, but only its rules.

C. Open Location Code

The use of latitude and longitude data provided by the GPS and the usage of Location Based Service (LBS) is fundamental to identify and represent a specific location. However, they are

considered challenging to use, since they require too much time to get an accurate position, and prone to errors, e.g., if swapped they represent a completely different position. A Location Encoding System can solve these problems by associating an alphanumeric, but short, string to a geographic location. More specifically, the Open Location Code (OLC) is an implementation of this type of system. It is a technology developed by Google that implements a partitioning of the whole Earth’s surface in “tiles”, then each tile is labeled using an unique code. The OLC is represented as a string from 2 to 15 characters long. The default OLC length is 10 digits and its precision is 13.9 meters. The higher the number of digits, the smaller is the considered area and the higher the precision of the location.

D. Decentralized Identity

Decentralized Identifiers (DIDs) are a new type of digital and globally unique identifier, standardized by W3C. Each ID is structured as a document (DID document) that contains detailed information about the DID itself. For instance, it can specify which is the entity that has the authority to modify the document (DID controller) and information for authenticating the DID owner. Moreover, it contains information about where to find the document (DID resolution) and the DID method that identifies where such resolution happens, e.g., *did:btc* specifies that the ID refers to the Bitcoin blockchain [12].

E. Related Works on Proof of Location

The majority of works on PoL focus mainly on security and privacy challenges. Generally, PoL systems rely on centralized verification approaches, based on the presence of a central database and related service that can check the proximity of a prover to a witness. Related works can be broadly subdivided into two categories, i.e. infrastructure dependent and infrastructure independent.

1) *Infrastructure dependent*: In this type of system, usually, a trusted fixed access point (e.g. Wi-Fi) or specific hardware is employed to check users’ location and then issues a location proof. In [13], for instance, users can communicate with access points or cell towers, requesting a LP. However, the privacy issues are not considered. Similarly, in [14], a centralized solution is proposed for generating LPs that manage the verification process through access points, which are considered trusted by default.

Another infrastructure-dependent project is FOAM [15] which is a decentralized open infrastructure that is trying to use the concept of zone and zone anchor (i.e. a radio beacon), placed outdoors, creating a trusted zone used for location tracking. FOAM uses Ethereum blockchain to allow users to contribute, and control when and with whom they share their personal data.

Another Ethereum-based project relies on the existing infrastructure of a mobile network operator leveraging a vast network of cell towers [16]. To apply this strategy, the user must be equipped with a terminal, or an IoT device, that must be locatable in terms of the network cells. Specifically,

the authors of the paper propose an approach to represent geofences using Ethereum smart contracts, checking if the user’s position (using the terminal) is located inside the virtual borders and then trigger specific actions accordingly. The drawback of this approach is the relevant cost that this solution requires.

2) *Infrastructure independent*: According to [17], decentralized systems can be used to design a more informed and participatory collective decision-making utilizing the concept of *witness presence*, and remain independent from the use of specific hardware/infrastructure. The idea is thus to resort to the presence of witnesses that use a short-range technology of wireless radios, e.g. Bluetooth, that ensures the physical proximity of mobile users nearby. The absence of access points allows a cheaper system deployment, with respect to one that requires a specific infrastructure. Usually, since the witness is not trusted, a Verifier user is required to verify the generated LPs. APPLAUS [4] is one of the pioneer infrastructure-independent projects that proposed a centralized scheme where, through a short-range communication method, users mutually generate location proofs and report them to a server. The proof requested by the prover is generated by a witness using a random number, pseudonym, computing a hash code and then signing it through a public-key encryption scheme. Then the proof will be sent to the central server by the prover. One of the participants in this architecture is the Central Authority, who knows the mapping between the public key and the real identity of the provers.

In the blockchain-based architecture proposed in [18], valid LPs are recorded into blocks. However, this solution is vulnerable to collusion attacks because the protocol allows direct communication between provers that could cheat the system.

Another infrastructure-independent and blockchain-based solution is the PASPORT architecture [19], where the main actors are the prover, witness, and verifier empowered to assign the witness to the prover for the location-proof generation. The authors claim that their system was Prover-Prover and Prover-Witness collusion resistant, however, the verifier could not act in “good-faith” and misbehave.

Gambis et al. proposed the PROPS architecture [20] which, although it follows a collaborative approach, uses a single Location Based Service for the verification phase.

III. DECENTRALIZED PROOF OF LOCATION SYSTEM

In this section, we describe the system architecture and the design choices we made to implement the PoL System and the related Decentralized Application (DApp).

The PoL DApp is able to retrieve data and related LPs and show them to the users without the need to interact with centralized authoritative servers. This is made possible through an (off-chain) interaction with a hypercube DHT for the data lookup and retrieval, while resorting to blockchain smart contracts to verify the truthfulness of the data in the DHT.

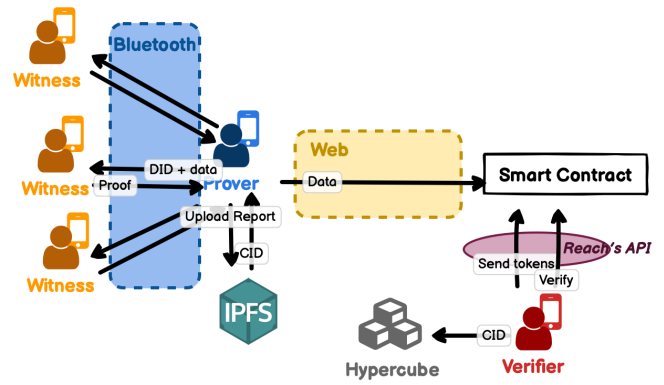


Fig. 1. High level description of the system architecture.

A. System Architecture

Figure 1 represents the general architecture of the system, characterized by the use of the following decentralized technologies, i.e. a Hypercube DHT for data location and retrieval [2], IPFS as decentralized file storage where data is actually stored [9], and the blockchain to run the PoL service.

Data storage and retrieval: As mentioned, data is stored in IPFS, thus the main issue here is how to retrieve data, once published in this decentralized file storage. While, in general, the Hypercube DHT might be used to perform data lookup based on a variety of possible keywords [2], in this work we focus on the location of generated contents. Thus, we are considering a typical situation where a user looks for some data which has been generated at a given location, i.e. the location is the keyword used to search contents. To this aim, the keyword set of the Hypercube is associated with the user’s location using the Open Location Code (OLC) technology [21]. Going into the specifics, a dual encoding is exploited. The original location of the user, represented with a classed latitude-longitude format, is firstly converted into the respective OLC and then to a *r-bit* string compatible to the ID space of the Hypercube DHT. This way, the location is associated with a node of the Hypercube DHT, which is the one entitled to handle information related to that specific geographical area. Details about this procedure can be found in [2].

The blockchain as the driver for PoL: The main part of the PoL system relies on the use of a smart-contract based blockchain technology. The main actors involved in the interactions are essentially four: the Prover, the Witness, the Verifier, and the Certification Authority. We describe here below their role in isolation.

- *Prover*. The Prover is a user with a mobile device, who needs to validate his/her location by obtaining verifiable proof.
- *Witness*. The Witness is a user located nearby the Prover. Its primary role is to compute and issue a LP that it will be sent back to the Prover. We assume Witnesses are untrusted. To ensure that LP cannot be forged, it is

signed by the private key of the witness generating the proof.

- *Verifier*. The Verifier is in charge of confirming the locations (and other information) stored in the blockchain. It also checks the validity of prover and witness signatures.
- *Certification Authority*. Another important actor is the Certification Authority (CA), not shown in Figure 1 for the sake of simplicity. The CA is in charge of certifying the Verifiers and maintaining the mapping between users' real identity and their pseudonyms (public keys). To become a Witness, one must authenticate to the CA using their public key. In this way, the witness's public key will be added to a *witnesses list*, which is delivered to the Verifier (this is a common solution exploited in the state of the art, e.g., PASPORT [19]).

B. Compute and verify the location-proof

Create the location-proof: When we talk about computing the LP, we are referring to the moment when the witness uses his private key applied to a hash function on Prover proof. In our work, the Prover will broadcast, to a nearby Witness, a request, Figure 2, that is composed of the following data:

- The current location;
- His DID;
- The nonce: a random number;
- The CID used to retrieve the report with IPFS¹.

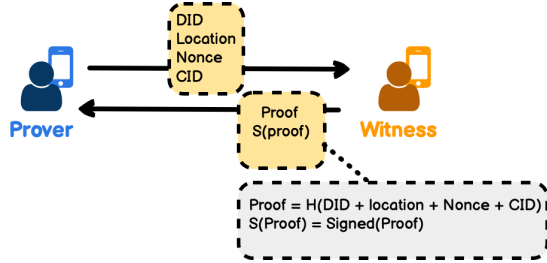


Fig. 2. Creation of a location-proof.

The data located inside the request will be used by the Witness to build a LP, signed with its private key, $S(Proof)$, and then send it back to the Prover. Recall that, before starting to build the LP, the Witness must authenticate the Prover using the DID resolution.

All of the parameters contained inside the request, shown in Figure 2, will allow the Verifier to check that the proof/certificate is associated with a well-identified Prover in a specific location, in a range of some witnesses. This is why we decided to hash the DID together with the location: if we hadn't put this, the Verifier wouldn't be able to attest that the Prover was in the position that he claimed.

Location-proof verification: After computing the LP, the data inserted by Provers inside the smart contract have to be verified by specific users, i.e. Verifiers, that will subsequently insert the data inside the hypercube.

¹We assume that the Prover has already uploaded his report's data on IPFS, thus obtaining the resulting CID.

In our approach, the Verifier has access to a list of public keys of Witnesses. This list will be delivered to Verifiers by the CA every time a new Witness is added, and will be used to check which Witness has signed the LP of the Prover.

Generally, the verification process executed by the Verifier is composed of two main parts:

- 1) Check that the proof is valid and has been signed by a known Witness;
- 2) Check that the hash inside the contract, signed by the Witness, is equal to the hash of the concatenation of DID, location, nonce, and CID. If their equality are confirmed, both the location and the CID correspond to the original declared by the Prover and attested by the Witness through the generation of the proof.

Specifically, the Verifier will check that the hash located inside the smart contract is equal to the hash result obtained by applying the public key of the witness on the signed proof.

IV. SMART CONTRACT BASED PROOF OF LOCATION

In this work, we leverage *smart contracts* to temporarily store the data that still have to be verified and which are sent by Provers. Specifically, we decided to associate a different location for every smart contract that could be created, storing the *contract id* inside the hypercube. In particular, some data that has to be stored are the following:

- DID of the prover;
- Hash of the Proof;
- Signed Hash of the proof;
- Prover wallet address: used to return possible rewards;
- The nonce used by the witness for generating the proof;
- CID (Content Identifier) of the data.

In order to store data inside the contract, Reach's Maps have been used. They allow the storage of information using a *key-value* approach. For the sake of scalability, every smart contract is in charge to a specific location area, whose size should be tuned based on different parameters such as the data workload overhead, geographical constraints, etc. To retrieve the smart contract the decentralized discovery system proposed in [2] is employed (but in general any kind of service discovery service can be used). The search key is the location area, the result of which is the smart contract in charge of that location.

A key aspect, during the bootstrap phase of the system, is the creation of the smart contracts associated with each specific area. To this extent, we adopt a solution based on a factory pattern. If a request for a novel area to be covered comes to the system, this request triggers the creation of a novel smart contract, created by a dedicated factory smart contract. This solution allows the users to trust a single smart contract, i.e. the factory, and its source code in charge of deploying novel smart contracts dedicated to the area. As many smart contracts will be deployed, the use of the factory pattern avoids the risk of the code being maliciously modified over time.

A. User rewards

The usage of smart contract enables an automatic reward to those nodes that participate to the PoL service. Several reward-

ing schemes might be devised, based on the type of specific application service. In our prototype, rewards are networks tokens, such as Algos or Ethers, that will be transferred by the smart contract to Provers in an automatic way when specific conditions happen. Indeed, in order to validate new proof of locations, the Verifier will have to insert a specific amount of reward tokens inside one or more contracts. A Prover will receive his rewards solely if the Verifier verifies his data, such as the LP, and inserts them inside the hypercube. The purpose of this type of incentive is to guarantee continuous participation in the project through the insertion of new reports.

V. EXPERIMENTAL EVALUATION

In this section, we will conduct a performance analysis of our architecture. In this work, we specifically focus on the interaction between the users and the smart contract. Indeed, some parts concerning the uploading of the data on Hypercube and IPFS have already been tested in similar application scenarios, thus demonstrating the viability of those solutions [2].

The Algorand and Ethereum performances' systems have been measured focusing on transaction latency and cost, both for the deploy and for the attach operations. In particular, it is worth noting that the deploy function not only is in charge of creating and deploying the smart contract in the blockchain, but it also inserts novel data into the system, such as the location and the creator's identity.

As already mentioned, the smart contract was implemented in Reach. We thus decided to test the smart contract over the Ethereum testnet, Goerli, the Polygon Mumbai testnet, and the Algorand testnet. This analysis was conducted using different numbers of users (up to 32) and creating the corresponding numbers of smart contracts, up to 8 (a smart contract can contain up to 4 users that attach to it, including the creator). It was not possible to extend the tests to a higher number of users. The reason is that each user needs a minimum amount of cryptocurrency tokens as a balance, in order to pay transaction fees, and even if testnets were used, it was not possible to obtain a sufficient amount of tokens to further increase the maximum number of users. Users' behavior was simulated through a set of digital agents, issuing some randomly generated synthetic data.

A. Results

Figures 3 and 4 show the latencies (in seconds) for the three testnets we used, respectively for the deploy and attach operations. In particular, Figure 3 shows the mean time (and standard deviation) experienced to create the smart contract and subsequently insert the data of the LP (location, creator DID), when varying the number of users that create the contracts. If we consider only the average time, we can notice that Polygon performs better than the other two blockchains. However, Algorand has a similar average latency and a noticeably lower standard deviation.

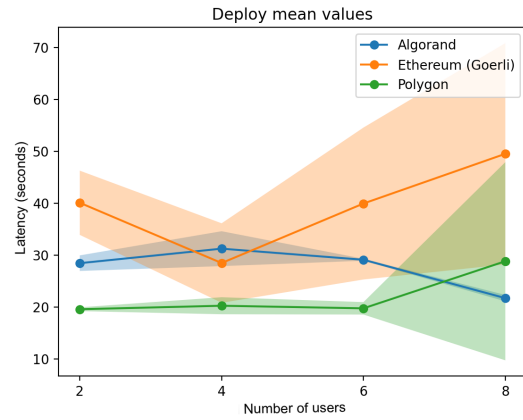


Fig. 3. Deploy mean latencies.

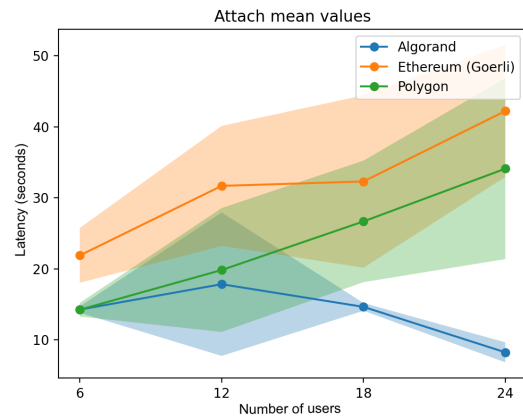


Fig. 4. Attach mean latencies.

For what concerns the attach operation, shown in Figure 4, Algorand confirms itself as the most performing blockchain, since its average latency is minimal and its standard deviation is lower than the other two networks. In this case, we used a different number of users, acting as Provers and interacting with the smart contract, in a range from 6 up to 24. The worse results obtained by Ethereum and Polygon can probably be explained by their specific protocol to add transactions in the ledger. In fact, the transaction throughput of these blockchain can significantly decrease in case of congestion.

Figures 5 and 6 show the box plot chart of the three testnets, considering the deploy and the attach. Generally, the creation of a contract can require more time than the attach phase, and this can be seen on every network. Again, what emerges is that Algorand performs better than Ethereum and Polygon, in terms of latency.

Another point in favor of Algorand is that its costs are very low as opposed to Ethereum and Polygon. A deploy operation on Goerli can reach a cost equal to 0.1401 ETH which, at the moment of writing (October 2022), corresponds to €171.18. Going into details, both Goerli and Polygon, have a deployment process that used 1,440,385 gas while the

amount of gas used for the attach is 82,437.

In conclusion, interacting with a smart contract that is deployed on Algorand could allow for saving a lot of time and money, and developing efficient applications.

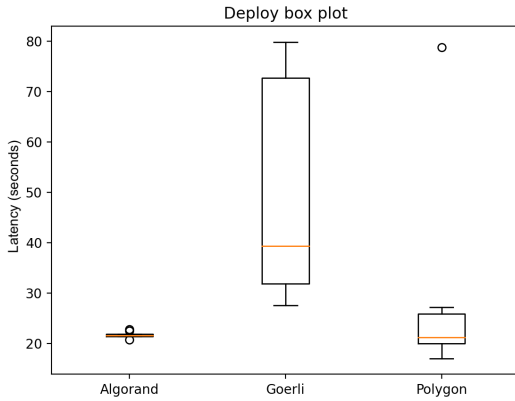


Fig. 5. Box plot of latency for the deploy operation.

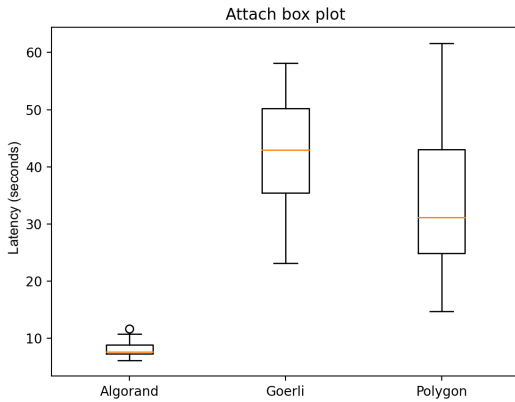


Fig. 6. Box plot of latency for the attach operation.

VI. CONCLUSIONS

In this paper, we proposed a decentralized architecture for PoL services. One of the primary roles of the proposed PoL architecture is to avoid fake data insertions (in terms of geographical provenance). Without it, users would be able to insert any kind of data, truthful or not, also with the possibility of overloading the system. This led us to identify two main challenges of our PoL system: i) computing the LP in a decentralized manner, ii) verifying that data inserted by Provers inside Smart Contracts are truthful LPs. Although the first point has been built to maintain its decentralization, the second is not completely decentralized for two reasons: there is a CA and not everybody can act as a Verifier.

An advantage of our decentralized architecture is that there is no need to employ trusted access points to generate new LPs. However, we did not focus on *Prover-Prover* or *Prover-Witness* collusions. This can represent future work.

REFERENCES

- [1] C. Prandi, S. Mirri, S. Ferretti, and P. Salomoni, "On the need of trustworthy sensing and crowdsourcing for urban accessibility in smart city," *ACM Transactions on Internet Technology*, vol. 18, no. 1, 2017.
- [2] M. Zichichi, L. Serena, S. Ferretti, and G. D'Angelo, "Complex queries over decentralised systems for geodata retrieval," *IET Networks*, 2022.
- [3] W. Lv, S. Wu, C. Jiang, Y. Cui, X. Qiu, and Y. Zhang, "Towards large-scale and privacy-preserving contact tracing in covid-19 pandemic: A blockchain perspective," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 282–298, 2022.
- [4] Z. Zhu and G. Cao, "Applaus: A privacy-preserving location proof updating system for location-based services," in *2011 Proceedings IEEE INFOCOM*, pp. 1889–1897, 2011.
- [5] G. Kondova and J. Erbguth, "Self-sovereign identity on public blockchains and the gdpr," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 342–345, 2020.
- [6] M. Zichichi, S. Ferretti, G. D'Angelo, and V. Rodríguez-Doncel, "Data governance through a multi-dlt architecture in view of the gdpr," *Cluster Computing*, vol. 25, no. 6, p. 4515 – 4542, 2022.
- [7] M. Zichichi, S. Ferretti, and G. D'Angelo, "On the efficiency of decentralized file storage for personal information management systems," in *Proc. of the 2nd International Workshop on Social (Media) Sensing, co-located with 25th IEEE Symposium on Computers and Communications 2020 (ISCC2020)*, pp. 1–6, IEEE, 2020.
- [8] J. Benet, "Ipfis-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.
- [9] M. Zichichi, S. Ferretti, and G. D'Angelo, "A distributed ledger based infrastructure for smart transportation system and social good," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, IEEE, 2020.
- [10] M. Zichichi, L. Serena, S. Ferretti, and G. D'Angelo, "Governing decentralized complex queries through a dao," in *Proceedings of the Conference on Information Technology for Social Good*, pp. 121–126, 2021.
- [11] "Reach language." <https://reach.sh>.
- [12] M.-H. Rhie, K.-H. Kim, D. Hwang, and K.-H. Kim, "Vulnerability analysis of did document's updating process in the decentralized identifier systems," in *2021 International Conference on Information Networking (ICOIN)*, pp. 517–520, 2021.
- [13] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, pp. 1–6, 2009.
- [14] C. Javali, G. Revadigar, K. B. Rasmussen, W. Hu, and S. Jha, "I am alice, i was in wonderland: secure location proof generation and verification protocol," in *2016 IEEE 41st conference on local computer networks (LCN)*, pp. 477–485, IEEE, 2016.
- [15] "Foam whitepaper." https://foam.space/publicAssets/FOAM_Whitepaper.pdf. [Online].
- [16] F. Victor and S. Zickau, "Geofences on the blockchain: Enabling decentralized location-based services," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 97–104, IEEE, 2018.
- [17] E. Pourmaras, "Proof of witness presence: blockchain consensus for augmented democracy in smart cities," *Journal of Parallel and Distributed Computing*, vol. 145, pp. 160–175, 2020.
- [18] M. Amoretti, G. Brambilla, F. Medioli, and F. Zanichelli, "Blockchain-based proof of location," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 146–153, 2018.
- [19] M. R. Nosouhi, K. Sood, S. Yu, M. Grobler, and J. Zhang, "Pasport: A secure and private location proof generation and verification framework," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 2, pp. 293–307, 2020.
- [20] S. Gambs, M.-O. Killijian, M. Roy, and M. Traoré, "Proprs: A privacy-preserving location proof system," in *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*, pp. 1–10, IEEE, 2014.
- [21] "Introduction to Open Location Code." <http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/>. [Online].