A Robust Spatio-Temporal Verification Protocol for Blockchain

Bulat Nasrulin[1], Muhammad Muzammal[2,3], and Qiang Qu[2(✉)]

[1] Shenzhen College of Advanced Technology,
University of Chinese Academy of Sciences, Shenzhen, China
bulat@siat.ac.cn
[2] Shenzhen Institutes of Advanced Technology,
Chinese Academy of Sciences, Shenzhen, China
{muzammal,qiang}@siat.ac.cn
[3] Department of Computer Science, Bahria University, Islamabad, Pakistan
muzammal@bui.edu.pk

Abstract. Massive Spatio-temporal data is increasingly collected in a variety of domains including supply chain. The authenticity as well as the security of such data is usually a concern due to the requirement of trust in centralised systems. Blockchain technology has come to the forth recently and offers ways for trustless and reliable storage and processing of data. However, current blockchain proposals either do not support spatial data or make simplifying assumptions such as 'trusted' servers to process spatio-temporal data. We presume that the notion of 'trust' in a blockchain is too strong an assumption and propose a robust spatio-temporal verification protocol for the blockchain. In this work, we present a novel practical proof-of-location protocol on top of a permissioned blockchain. The protocol is instrumented by the implementation of an access control model and utilises a set of verification rules to create and verify spatio-temporal data points. We also propose a threat-to-validity model to evaluate the robustness of the verification protocol. The applicability and practicality of the protocol is demonstrated by the implementation of a supply chain case study as a proof-of-concept.

Keywords: Proof of location · Spatio-temporal consensus

1 Introduction

Many real-life applications collect spatio-temporal data for providing location-based services [8], for example, location-based web search, real-time traffic updates, nearest fuel station, and popular food points. Some applications offer incentives to the users [14] in the form of discounts or e-coupons for sharing location. One of the limitations of such applications is the assumption of 'trust' that the location shared by a client, for instance, an online check-in [13], is correct.

In some situations, it is important that the user data is accessible only if the presence of the user at a specific location is verified. For example, user data

H. Hacid et al. (Eds.): WISE 2018, LNCS 11233, pp. 52–67, 2018.
https://doi.org/10.1007/978-3-030-02922-7_4

is accessible to the physician only if the user is present at that time [19], or special kinds of certificates can be used only in limited locations. A number of application scenarios exist where a proof-of-location (PoL) is a requirement, for instance, to verify the authenticity of a financial transaction, to prevent impersonation attacks, and to reduce the possibility of spam and DOS attacks.

However, the implementation of a reliable location-based service is not obvious due to the requirements of (i) privacy and security, and (ii) integrity of data. An inherent limitation in today's centralised business solutions is the notion of 'trust' and the encryption mechanisms in centralised business solutions do not always guarantee data security as when such a system is compromised, sensitive user data is revealed. Similarly, business processes such as supply chain have a complex eco-system [17], for instance, in a minimalist settings, the stakeholders in a supply chain include sets of producers, logistic operators, distributors, and consumers. However, current supply chain solutions [2] lack transaction transparency due to trust-based centralised solutions.

Recently, the interest in the Blockchain systems and the advancements in Distributed Ledger Technology (DLT) have enabled 'trustless' reliable decentralised applications [18]. In short, a blockchain is a decentralised linked data structure with a state transition machine replication. Current blockchain systems lack the support for location-based services as the locations reported by the users through trusted servers are considered legitimate. However, the incentives associated with the locations [21] encourage false location reporting and issues such as GPS or Wifi spoofing [7] and colluding location attacks [22] are common.

1.1 Motivation

Consider a supply chain scenario where an item is transferred from a producer to a consumer by way of a transportation agent. The transportation of the product is of interest to the end-user, for example, the transportation time. However, the agent may wish to delay the shipment due to reasons such as putting multiple long-distance shipments together and thus, may attempt to report a location which is not true. Typically, special digital certificates are used for validating PoL. Many digital certificate-based solutions to attest the spatio-temporal location of an object have been proposed [6,20]. However, current PoL systems lack robustness as they require trusted centralised servers for operation and are vulnerable to various privacy and security attacks. Therefore, a robust and reliable trustless PoL protocol is desired that ensures the integrity and security of the location data.

1.2 Our Contribution

In this paper, we present a robust and reliable proof-of-location protocol that is implemented using a permissioned blockchain and ensures the data security

and integrity. The proposed system enables a reliable supply chain without the involvement of trusted third parties. Our contributions are as follows:

(1) We formalise the spatio-temporal and security requirements for a decentralised PoL protocol using digital certificates.
(2) We propose a practical robust PoL protocol that is trustless, decentralised and ensures the creation and validation of proof-of-location efficiently.
(3) We propose and evaluate the threats-to-validity for the protocol.
(4) We implement a proof-of-concept supply chain business case based on Hyperledger Iroha to demonstrate the applicability of the protocol.

Significance of the PoL Protocol. Many PoL protocols have been proposed in literature. This study formalises the problem (Sect. 3) by proposing the necessary and sufficient conditions that should be satisfied for a PoL protocol. A threats-to-validity model (Sect. 3.2) lists the possible threats to the protocol and the robustness of the protocol is demonstrated for the possible threats (Sect. 5).

The rest of this paper is organised as follows. In Sect. 2, we discuss related concepts including centralised approaches for achieving location proofs, previous attempts for spatio-temporal blockchains and location attacks. We formalise the problem in Sect. 3 and present the PoL protocol in Sect. 4. Threats-to-validity are evaluated in Sect. 5 and a proof-of-concept supply chain case study is presented in Sect. 6. Section 7 concludes this work.

2 Related Work

Large volumes of spatio-temporal data are being collected by Smart devices using location-based services. However, a device may not report the correct location due to reasons such as privacy concerns. The authenticity of location is established by techniques similar to location proofs [21]. A number of studies [8,20] have considered secure privacy-aware location proofs. Studies in literature can broadly be distinguished into infrastructure-based and infrastructure-independent location-proof studies. Infrastructure-based location verification assumes the presence of trusted access points, typically Wifi points, or other short-range communication media [9,15]. The location reported by such mechanisms is although correct, it has some limitations. For example, access points are hard to scale due to limited coverage. The study [8] considers wireless proofs using spatio-temporal properties of wireless channels whereas SecureRun [14] utilises wifi access points to get PoL.

Solutions for infrastructure-independent location verification are by way of a short-range communication between collocated smart devices. This approach mitigates the cost of deploying infrastructure but the proof generation is harder to create and is less secure. Moreover, users need to manually maintain a connection with nearby devices. Many studies [13,20] have considered similar ideas for the PoL. It should be clear that both classes of solutions assume a centralised trusted server to store PoL. A distinction is a protocol *PROPS* [6] that stores

location proofs on a personal smart device. But this approach hardens spatio-temporal analytics, as user is required to be online. Another interesting proposal is the use of internet [1] for long-range location verification with the assumption that the protocol is resistant to proxy or relay attacks.

Some initial considerations for reliable blockchain PoL include Ethereum smart contracts *Sikorka*[1] and location-on-blockchain [3]. However, these proposals do not elaborate on practical details and security discussion. Further, public blockchains that use Nakamoto consensus [11] lack a transaction finality which is crucial for industrial applications such as supply chain. Currently, two projects, FOAM[2] and XYO[3], are under development that provide Ethereum smart contract-based reliable spatio-temporal information. The main idea of FOAM protocol is to create a spatial-index with crypto-spatial coordinates by the assignment of an unforgeable index to each object. However, the requirement of special hardware deployment is a major drawback. XYO Network, in contrast, is based on the assumption of a shared network where each participant has a clear role and gets incentives for forwarding, collecting or storing location data [10,12].

3 Proof-of-Location Problem

The objective of a PoL protocol is to attain failure-tolerance in a decentralised environment. Proof-of-location is generated via the interaction of multiple untrusted or semi-trusted parties including validators, provers, witnesses and network peers. In order to achieve security for a given region, a set of witnesses must be deployed. It has been shown that a total of $3f + 1$ participants can only tolerate f Byzantine failures [4]. This is impractical as it requires massive hardware deployment. We now formally define the problem.

Definition 1 (Proof-of-Location). *A proof-of-location is a verifiable digital certificate that attests the presence of a prover σ at location l and time t and is signed by an authorised witness ω.*

Following set of properties are desirable for a reliable PoL protocol, assuming the presence of malicious actors [6]:

Definition 2 (Completness). *A PoL is considered complete if σ is attested at l, at time t, by $\omega \in W$, such that ω is registered at location l.*

Definition 3 (Spatio-Temporal Soundness). *A PoL is spatio-temporal sound if σ is not able to obtain a proof when not physically present at $\{l, t\}$.*

Definition 4 (Non-Transferability). *A non-transferable PoL attesting the location of a σ is valid only if it was provided by a prover $\sigma \in S$.*

[1] Online at http://sikorka.io.
[2] Online at https://foam.space.
[3] Online at https://xyo.network/whitepaper.

Definition 5 (Tamper-Evidence). *A PoL is tamper-evident if tampering can be detected.*

Definition 6 (Proof-of-Location Problem). *A PoL is secure if it is complete, spatio-temporally sound, non-transferable and tamper-evident.*

Table 1. Table with useful notations

Notation	Meaning
G	Block transaction set
$P = [\gamma_1 \dots \gamma_n]$	Network peers set
$S = [\sigma_1 \dots \sigma_k]$	Prover set
$W = [\omega_1 \dots \omega_m]$	Witness set
Z	Proposal transaction set
f	Number of Byzantine failures of system entities
l	Reported location
t	Transaction timestamp
$\delta; \Delta$	Minimum network delay; network imprecision
θ	Proximity threshold
τ	Signed transaction
ζ	Maximum proposal threshold
ρ	Dynamic entity
κ_s, κ_p	Cryptographic keypair: secret key, public key

In addition to PoL, a peer-to-peer system must provide (i) security, (ii) liveness, and (iii) fault-tolerance. Note that (i)–(iii) are not simultaneously achieved [5], as (i) security requires all results to be valid for the network, (ii) liveness guarantees termination, and (iii) fault-tolerance requires a result regardless of the network failure (Table 1).

3.1 Protocol Entities

The entities can either be static or dynamic. Dynamic entities do not have a fixed location recorded in the blockchain and move in a limited geographical region and communicate with nearby witnesses. It is usually assumed that the movement is initiated by way of a smart-contract that defines the starting and ending points, e.g. an object transfer from entity A to entity B. Static entities, on the other hand, are associated with some specific location that is verified and recorded in a blockchain. Examples of such entities include producers, like farmers and regular customers, waiting for delivery at predefined locations. We consider the following static entities for the proposed protocol.

Witness. A witness is a static entity and is associated with a particular location which is agreed and stored by network peers. Communications are performed

using a small communication delay that allows preventing proxy or relay attacks using delay-based challenge-response scheme. We assume that witness is able to perform cryptographic operations such as checking the validity of signatures and sign messages. Thus, the correctness of data is ensured by the witnesses as they provide data security and protection against the location attacks. We assume that there exist multiple witnesses such that the number of colluding malicious witnesses does not exceed a pre-defined threshold f_ω.

Prover. A prover communicates with nearby witnesses only if they are at a distance lower than a proximity threshold θ. We assume that a registered prover has a public/private keypair and the public key of the prover is known to all network peers as a pseudonym identifier. Prover can be either dynamic or static. When a dynamic entity is moving near a witness it performs a challenge-response authentication to collect location proofs. Similar authentication is performed for a static entity that desires to join the protocol.

Network peer. Transaction history is maintained by network peers in a blockchain. Blockchain supports identity mapping to a location, and the authenticity of the data in presence of adversaries. The network peers not only maintain system integrity but also provide PoL. Each peer is open for queries from witnesses, clients and other peers. Smart contracts are generated and executed as system transactions, for example, a transfer of goods from one owner to another. These smart contracts serve as a proof of an intent or an interaction between parties and are used during the validation process.

Client. A dynamic entity that moves in a restricted area is known to the client and the client makes queries to get the location or path of a dynamic entity at a specific time.

3.2 Threat Model

In order to create a PoL, a witness must be registered and verified at location (x, y). A witness performs proximity tests at a location within a maximum radius θ around a point (x, y). To tolerate f Byzantine failures $3f + 1$ nodes must be deployed [4]. This is a known lowest upper bound to achieve security in an asynchronous peer-to-peer network with possible Byzantine failures. We say that the protocol can tolerate up-to f_γ peer failures and f_ω witness failures. Users are registered with the network peers, and network peers maintain a list of users, witnesses and associated locations. The users and the witnesses are recognised by their public keys by the network peers. We assume that the public-private key pair is unique and is stored securely on personal devices.

For the threat model, we consider the following[4]: (i) a malicious prover submits a false location claim without being physically present at a location, (ii) a prover or network peer attempts to perform a colluding attack in order to manipulate an old or new location proof, (iii) a malicious witness either reports about a prover that is not present in close proximity, or does not report a prover

[4] We do not consider user privacy issues in this study and consider it as a future work.

presence that is in close proximity, and (iv) an adversary attempts a Sybil attack by way of creating multiple malicious users, peers or witnesses.

4 Proof-of-Location Protocol Design

We now present a protocol design that satisfies the conditions (Sect. 3) for a secure and reliable location tracking. We consider a peer-to-peer network where location proofs are generated by witnesses and are sent to peers for verification and storage. Peers exchange transactions and agree on the validity of a location proof by way of a consensus routine. An illustration of the PoL protocol design is presented in Fig. 1. We segregate between commands and queries to improve the performance of the system. A query is sent only to one peer which returns query results and a verification object that validates the query result. A command is packaged as a transaction and is put into the execution pipeline of the blockchain. Transaction validation is both via stateless and stateful validation. Stateless validation is for signature and time validation. Stateful validation is by way of a 'is_valid' predicate which validates transactions according to the system rules relative to the current local state. For example, the requirement of balance validation for an asset transfer.

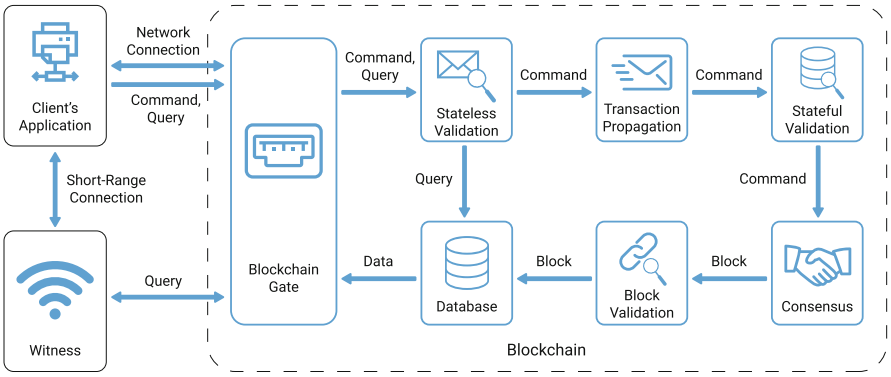


Fig. 1. An illustration of PoL protocol design

4.1 Proof-of-Location Consensus Algorithm

An outline of the consensus algorithm is given in Algorithm 1. The connection with a blockchain peer is established through a blockchain gate which is a network connection between a client and blockchain peer. Transactions are included in a proposal and the most popular proposal is decided by the consensus algorithm as a new block. Next, the block is validated and applied to the local database. The pipeline of the consensus algorithm is as follows: transaction is sent to the ordering peers (line 1), transactions are packaged into a proposal (line 3–7), proposal is shared with validating peers (line 8–9), the most popular

Algorithm 1. An outline of the spatio-temporal consensus algorithm

```
1  function RecieveTransaction(τ: Transaction; P = [γ1 ... γn]: Network peers;
   ζ: Maxmimum proposal size; tZ: Proposal time delay)
2      Z ← ∅                                                  ▷ Proposal set
3      for all unprocessed transaction τ do                   ▷ Process transaction queue
4          if StatelessValidation(τ, Sender(τ), Signature(τ)) then
5              Z ← τ ∪ Z
6      if |Z| < ζ then
7          Delay(tZ)
8      Send proposal Z to peers [γ1 ... γn] ∈ P

9  function ProcessProposal(Z: proposal; P = [γ1 ... γn];)
10     Zv ← StatefulValidation(State(γi), Z)                  ▷ Verified proposal
11     Λ ← Leader(Zv, State(γi))                              ▷ Decide leader on proposal and state
12     Send Vote(VP), sig_γi(VP) to leader Λ

13 function ProcessVote([hk, sig_γj]: peer vote; P = [γ1 ... γn]: Network peers;)
14     Vote hk, sig_γj received from γj
15     if γj ∈ P ∧ sig_γj ∈ State(γi) then                    ▷ Peer γj is known to peer γi
16         votes[hk]= votes[hk] ∪γi
17     for all hash h ∈ votes do
18         if |votes[h]| ≥ 2fγ + 1 then
19             Send block with hash h to peers [γ1 ... γn] ∈ P

20 function ProcessBlock(G: block; P = [γ1 ... γn]: Network peers; peer_i: Current
   peer)
21     if BlockValidation(G, textscState(γi)) then
22         NewState =Apply(textscState(γi), G)
23         return NewState
24     NewState ← Synchronise(P)                              ▷ Synchronise state with peers
25     return NewState
```

proposal is decided with a leader-based consensus protocol (line 9–20) and is formed as a new block (line 20–21). A consensus process produces a validated block with signatures of peers agreeing on the validity of the block. The validity of the received blocks is decided with a block validation routine (line 23). Finally, the accepted block is applied to the local datastore (line 24).

Assume that ρ moves from A (i.e. l_A) to B (i.e. l_B). A transaction τ_s is issued that defines the movement, i.e. source, target, path, system and business rules.

When passing through some route, the dynamic entity ρ communicates with the witnesses $[\omega_1 \ldots \omega_n]$ with a location proof generation protocol. Each witness is bonded with at least one network peer that stores and transmits location proofs to other network peers. Peers ratify location proofs on their soundness and completeness. The final validity of a location proof is decided by the consensus mechanism (Algorithm 1). When ρ arrives at the target location, an ending

transaction τ_e is generated by ρ which is a confirmation of the execution of the terms as defined in τ_s, for example, an asset transfer between A and B.

Network Initialisation. The first block generated during network initialisation is the genesis block G_0 which lists ledger rules including validation rules, participants, roles, permissions and threat model. Thus, parameters such as the number of permitted peers and witness failures are set during initialisation.

New Entity Join. When a new system entity, such as a network node or a client, wants to join the network, the following is performed: entity generates a cryptographic key-pair (κ_s, κ_p) followed by the creation of a join transaction which is sent to the blockchain nodes. If the role requires fixing a static location, location proofs are collected from the nearby witnesses. The signed join transaction is updated to the datastore after consensus.

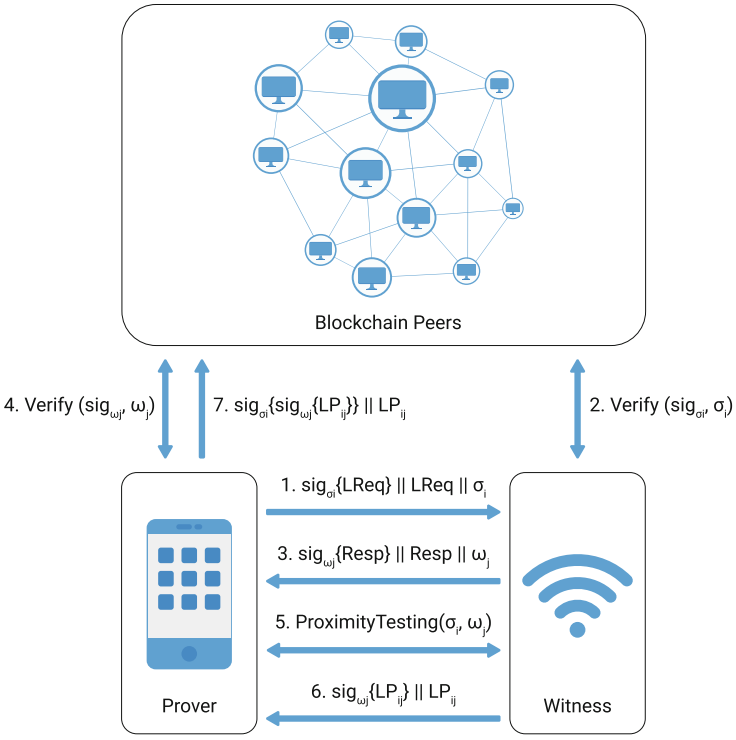


Fig. 2. An illustration of PoL generation

4.2 Proof-of-Location Generation

When a prover desires to obtain a proof of current location, the proof generation procedure is initiated in collaboration with the nearby witnesses. A short-range location proof is considered more reliable than a long-range location proof [16].

The proof generation protocol is illustrated in Fig. 2. The protocol consists of following phases:

(0) The prover σ_i is known to the blockchain peers due to the new entity join procedure. Next, σ_i prepares a list of witnesses in near proximity.
(1) Prover σ_i selects a witness ω_j and sends a location proof request. A message, $LReq||\sigma_i||sig_{\sigma_i}\{LReq\}$, is sent to witness ω_i. The message components include: $LReq$ to initialise the proximity testing, declared position $[x_i, y_i]$ at time t. Signature sig_{σ_i} and σ_i are used to authenticate the request.
(2) Witness processes the location proof request by validating the signature sig_{σ_i}, position $[x_i, y_i]$, timestamp t and the pseudonym σ_i. Position $[x_i, y_i]$ is validated relative to the proximity of the witness ω_j. Pseudonym σ_i must be known to the network peers and timestamp t must be within a valid range, i.e. $now - \delta \leq t \leq now + \delta + \Delta$, where δ is a minimum network delay, Δ is the network imprecision, now is the local clock time of the witness.
(3) Witness ω_j responses with a message $Resp$ whereas pseudonym ω_j and signature for $Resp$ message are used for the acknowledgement authentication.
(4) $Resp$ message from witness ω_j is validated with the help of network peers that includes the following checks: pseudonym ω_j is known to the peers, entity has a 'witness' role, the signature is valid and consistent with the pseudonym ω_j.
(5) γ_i and ω_j start a proximity testing protocol. Typically, witness and prover perform a fast bit exchange where witness sends some generated secret and waits for a response from the prover with the same secret. Each message is signed to protect against proxy attacks.
(6) After a set of rounds of proximity testing, witness is convinced about the declared location of γ_i. Witness sends signed location proof lp_{ij}, where lp contains declared location of entity γ_i, location of witness ω_j and current timestamp t_{ω_j}. lp_{ij} is sent to γ_i.
(7) Prover σ_i signs received location proof lp_{ij} and sends it to the network peers. Network peers share with each other the location proof and run consensus algorithm on the validity of the location proof.

Location Proof Verification. Location proof is considered valid if (i) it contains a valid timestamp, (ii) it is signed with at least $2f_\omega + 1$ witnesses and the prover, (iii) the witnesses and the prover are known, (iv) each witness is an authorised 'witness', and (v) all witnesses can validate the proximity of the prover in location $[x_i, y_i]$.

5 Threats-to-Validity Check

We now evaluate the threats-to-validity as we first discuss the adversary model.

5.1 Adversary Model

We distinguish among following adversary[5] types:

[5] We assume that an adversary can not violate cryptographic assumptions.

1. *Malicious prover.* An adversary tries to modify current real-time or position and prove the wrong location, or lie to the verifier about its position. A malicious prover attempts to (i) create a fake location proof, (ii) lie to the witness or the verifier, or (iii) steal a location proof of another user.
2. *Malicious verifier.* An adversary can approve fake location proof, or blacklist and censor transactions from a legitimate user.
3. *Malicious witness.* An adversary can try to foul a legitimate prover or verifier by a fake location proof. For example, proof of proximity of a physically-not-present or non-existing entity.
4. *Malicious network peer.* Malicious peer attempts to attack consensus process to include transactions that are not valid. Sybil attack and double spending are examples of such attacks.
5. *Colluding users.* A colluding attack with malicious parties is performed to accept and include a fake location proof. Colluding adversary attacks are considered hard to be detected and prevented [6].

5.2 Adversary Resistance

We now present the adversary resistance of the protocol according to the criteria discussed in Sect. 3.

Correctness. Assume that prover σ submits a proof of presence at location l and time t. The validity of the proof is decided by a consensus process. With the assumption that number of prover failures is less than f_γ and number of witness failures is less than f_ω, the protocol achieves completeness. Spatio-temporal soundness is achieved by validating the signatures and location information. A malicious prover may attempt a distance fraud. The proximity testing described in section 4.2 prevents a distance fraud.

Proof of Ownership. A proof of ownership for the location of prover σ requires the signature of σ which is not possible without the secret key κ_s^σ.

Authenticity. A change in the location proof invalidates the signatures which in turn makes location proof invalid. A location proof is stored in a blockchain which is maintained by network peers. As long as the number of malicious peers is less than f_σ, authenticity of the datastore is ensured.

Colluding Attacks. A fake location proof generated by a malicious witness with the help of a malicious prover is rejected if the number of witness signatures is less than $2f_\omega + 1$. Due to the assumption that the number of malicious witnesses is less than f_ω, it is not feasible to create a valid fake location proof.

6 Case Study: Supply Chain

A supply chain is a system of organisations, people, activities, information, and resources involved in moving a product or service from a set of suppliers to a set of customers. Typically, a supply chain can be represented as a sequence of

transitions from a producer to the consumer with multiple intermediate points. We consider the transportation chain as the communication chain between two parties. Let's call this communication between party A and party B. Party A is a service provider for some goods, party B is a customer.

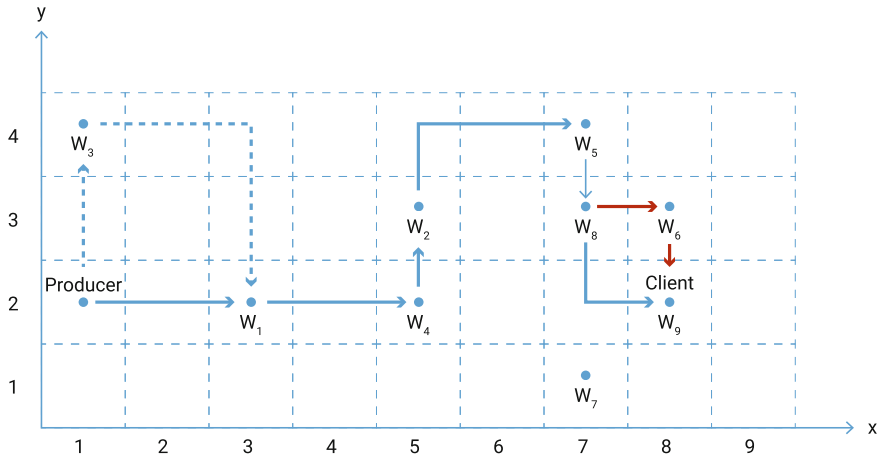


Fig. 3. Example Grid

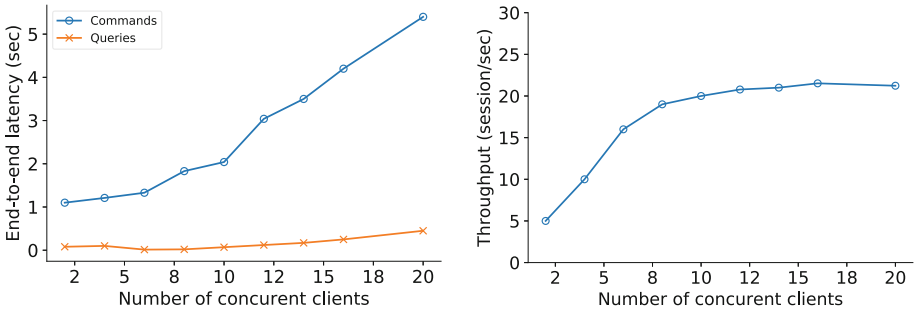
Consider a 4×9 grid as shown in Fig. 3 with many stakeholders in the chain. We say that, Party A is a producer and party B is a customer who is intending to buy some amount of goods in exchange of some assets. The blockchain records the fact that party A is the owner of the commodity and party B is the liquidity bearer. Party A and Party B are associated with a static location. For example, party A is a producer in location $[1, 2]$, and party B is a consumer in location $[8, 2]$. To deliver goods from one location to another, a transport is used which is a dynamic entity that delivers goods in exchange of some fee. Witnesses $\omega_1, \ldots, \omega_9$ are present in different locations to generate location proofs. The proof pipeline is as follows:

1. Party A and party B agree on terms of exchange via a starting transaction: product P is put to the transport T in exchange to the payment from party B at location $[1, 2]$ to party A at time t_1 in order to deliver it to the B at location $[8, 2]$.
2. The product is transported from location $[1, 2]$ to location $[8, 2]$.
3. Through the road, witnesses are tracking all bypassing transports, perform a proximity test and report to the blockchain peers.
4. The proof of location transaction is sent to the blockchain nodes, where each node is validating it and writing the results to the underlying data-store.
5. When transport arrives to $[8, 2]$ client confirms the delivery with an ending transaction: a product is delivered to location $[8, 2]$ at time t_2 and is verified

by party B. Ending transaction closes the delivery as it transfers the required number of assets to the producer, fixes the ownership of the product with B and pays transportation fee for the transport T.

6.1 Implementation Details

We implement a software solution such that we create both legitimate and malicious actors. Our implementation[6] is using Hyperledger Iroha[7] as an underlying blockchain platform. Hyperledger Iroha is an open-source, distributed ledger technology platform implemented in C++. The codebase used is the latest development branch. We deploy 4 virtual private servers. Each peer is deployed in a docker environment, having a docker image created from the development branch at a specified commit (`3404b17`), and a PostgreSQL 9.5 docker container for each Iroha docker container.



(a) Session latency (b) System throughput

Fig. 4. System latency and throughput analysis

A transaction is implemented as an asset transfer to an agreed transport. The consumer sends a transaction and locks the funds of the transport by turning transport into a multi-signature account with quorum 2. Producer transfers the ownership of a product, adds new signatory and increases the quorum of transport account. In the end, transport is a multi-signature account that requires at least three signatures for any further transactions. This ensures the safety of the funds before reaching the consumer. Upon arriving at the desired location, transport creates an ending transaction that transfers coins to the producer, product ownership to the consumer and transportation fees to the transporter wallets. The transaction is checked by three parties for agreed number of coins and products.

We simulate a supply chain case and deploy transportation units in a test area and show how it affects the performance of the system. A witness is capable

[6] https://github.com/grimadas/iroha-supply-chain.
[7] https://github.com/hyperledger/iroha.

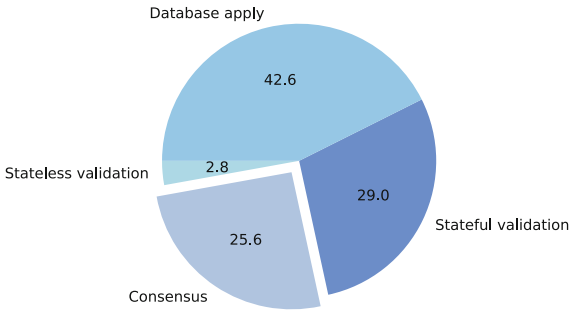


Fig. 5. Time distribution of a transaction at different execution stages

of attesting an area within one cell. An end-to-end transaction latency for the full pipeline for location proof transaction is shown in Fig. 4. The results show that with an increase in the number of concurrent clients generating and submitting a location proof, latency increases linearly. The system is capable of creating and validating 30 location proofs per second. The end-to-end latency increases linearly with the number of concurrent clients, while the latency of read-only queries changes insignificantly. As Iroha is still in development, a number of optimisations can be applied to improve the system performance. Figure 5 shows time distribution of different components. It can be seen that database apply operations are the most time-consuming.

7 Conclusion

In this paper, we propose a practical robust proof-of-location protocol on top of the blockchain. We implement a proof-of-concept supply chain and evaluate the security and the performance of the protocol in the presence of malicious actors. We illustrate in practice that system scales linearly with the increase of concurrent actors. To the best of our knowledge, ours is one of the first practical systems that achieves tamper-resistance for the accepted Proof-of-Location transactions. We show the utility of our solution in the supply chain domain. A dis-intermediation of actors with defined roles allows using both public and private domains.

A number of challenges still remain. For example, flexible privacy settings with zero-knowledge proofs and incentive design with game-theoretic guarantees. As a future work, we consider an extension of the protocol to work with dynamic witnesses and the deployment of the solution in real-world settings.

Acknowledgments. The work was partially supported by the CAS Pioneer Hundred Talents Program, China [grant number Y84402, 2017], and CAS President's International Fellowship Initiative, China [grant number 2018VTB0005, 2018].

References

1. Abdou, A.M., et al.: Location verification on the internet: towards enforcing location-aware access policies over internet clients. In: 2014 IEEE Conference on Communications and Network Security (CNS), pp. 175–183 (2014)
2. Ahi, P., Searcy, C.: Assessing sustainability in the supply chain: a triple bottom line approach. Appl. Math. Model. **39**(10–11), 2882–2896 (2015)
3. Brambilla, G., Amoretti, M., Zanichelli, F.: Using block chain for peer-to-peer proof-of-location. arXiv preprint arXiv:1607.00174 (2016)
4. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OSDI, vol. 99, pp. 173–186 (1999)
5. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM (JACM) **32**(2), 374–382 (1985)
6. Gambs, S., Killijian, M.O., Roy, M., Traoré, M.: PROPS: a PRivacy-preserving location proof system. In: IEEE 33rd International Symposium on SRDS, pp. 1–10 (2014)
7. Jafarnia-Jahromi, A., Broumandan, A., Nielsen, J., Lachapelle, G.: GPS vulnerability to spoofing threats and a review of antispoofing techniques. Int. J. Navig. Obs. (2012)
8. Javali, C., Revadigar, G., Rasmussen, K.B., Hu, W., Jha, S.: 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 (2016)
9. Muzammal, M., Gohar, M., Rahman, A.U., Qu, Q., et al.: Trajectory mining using uncertain sensor data. IEEE Access **6**, 4895–4903 (2018)
10. Muzammal, M.: Renovating blockchain with distributed databases: an open source system. Futur. Gener. Comput. Syst. **90**, 105–117 (2019)
11. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009). https://bitcoin.org/bitcoin.pdf
12. Nasrulin, B., Muzammal, M., Qu, Q.: ChainMOB: mobility analytics on blockchain. In: 19th IEEE International Conference on Mobile Data Management, MDM, pp. 292–293 (2018)
13. Ni, X., et al.: A mobile phone-based physical-social location proof system for mobile social network service. Secur. Commun. Netw. **9**(13), 1890–1904 (2016)
14. Pham, A., Huguenin, K., Bilogrevic, I., Dacosta, I., Hubaux, J.P.: SecureRun: cheat-proof and private summaries for location-based activities. IEEE Trans. Mob. Comput. **15**(8), 2109–2123 (2016)
15. Qu, Q., Liu, S., et al.: Efficient online summarization of large-scale dynamic networks. IEEE Trans. Knowl. Data Eng. **28**(12), 3231–3245 (2016)
16. Ranganathan, A., Capkun, S.: Are we really close? Verifying proximity in wireless systems. IEEE Secur. Priv. (2017)
17. Stadtler, H.: Supply chain management - an overview. In: Stadtler, H., Kilger, C. (eds.) Supply Chain Management and Advanced Planning, pp. 9–36. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-74512-9_2
18. Underwood, S.: Blockchain beyond bitcoin. Commun. ACM **59**(11), 15–17 (2016)
19. Wan, J., Zou, C., Ullah, S., Lai, C., Zhou, M., Wang, X.: Cloud-enabled wireless body area networks for pervasive healthcare. IEEE Network **27**(5), 56–61 (2013)
20. Wang, X., Pande, A.: STAMP: enabling privacy-preserving location proofs for mobile users. IEEE/ACM Trans. Netw. **24**(6), 3276–3289 (2016)

21. Waters, B., Felten, E.: Secure, private proofs of location. Department of Computer Science, Princeton University, Princeton, NJ, USA, Technical report (2003)
22. Yang, J., Chen, Y., Trappe, W., Cheng, J.: Detection and localization of multiple spoofing attackers in wireless networks. IEEE Trans. Parallel Distrib. Syst. **24**(1), 44–58 (2013)