Decentralised Land Registration and Transaction with Blockchain and Self-Sovereign Identity

Pedro C. Henriques Miguel Correia

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa – Lisboa, Portugal
{pedro.carvalho.henriques, miguel.p.correia}@tecnico.ulisboa.pt

***Abstract*—Land registry systems maintain public records of land ownership, transactions, and property boundaries, offering legal documentation and information on titles and associated rights. However, there are challenges in the security, accuracy, efficiency, and transparency of the current registries. The paper proposes the Decentralised Land Registry (DLR), a system that leverages a permissioned blockchain, Hyperledger Fabric, enabling a secure and controlled environment for land registration and transaction validation. The use of smart contracts ensures consensus and immutability and allows support for legitimate property transactions overseen by registry authorities. Decentralised identities and ownership proofs are issued using a Self-Sovereign Identity approach, leveraging Hyperledger Indy as the ledger and Hyperledger Aries as the agent representing government entities and issuing these credentials. This solution would mark a pivotal shift in land registry systems, revolutionising traditional practices by addressing their limitations and fostering a transparent, reliable, and efficient landscape for managing property rights and transactions.**

I. Introduction

Land registry systems are the backbone of a country's land ownership infrastructure. Given that land is a valuable asset, secure, accurate and up-to-date records are critical. Currently, land registry systems are viewed as complex and vulnerable, packed with inefficiencies, and susceptible to fraudulent activity. One of the main problems of these systems lies in the methods used to record and manage land ownership. These usually involve paper documents which, even if digitalised, are not in a manageable and semantic format, causing constraints when trying to provide accessible and up-to-date land information and also discrepancies in record-keeping [1]. Moreover, centralised databases create single points of failure, increasing the risks of unauthorised access, tampering, and loss, while also raising concerns about transparency, which can lead to disputes and prolonged transaction times [2].

Blockchain technology enables transforming current systems into transparent, efficient, and highly secure platforms. Using distributed ledgers, it is possible to issue and store ownership proofs in a secure manner, across a network of nodes that prevent changes to existing data. Furthermore, since blockchains are immutable, a system with these characteristics creates an authentic and accurate record of ownership that users can trust. As ownership proofs can be created and emitted using a predefined and manageable format, the ease of tracking and updating data increases drastically. Through smart contracts, which are self-executing contracts that are programmed to depend on certain conditions and enforce and execute actions, the number of intermediaries can be reduced. In a system like this, users need to have their identity verified, which is also possible through blockchain, leveraging Self-Sovereign Identity (SSI) solutions [3]. In this way, individual decentralised identities can be created, ensuring that only the real owner of an identity can execute actions on its behalf.

This paper presents the *Decentralised Land Registry* (DLR), the first land registry system proposal that brings together the following characteristics. DLR is a decentralised application on a permissioned blockchain, ensuring land registry and transaction efficiency while guaranteeing integrity, confidentiality, authenticity, and transparency; DLR leverages decentralised identity to provide users with full control over their identity, increasing privacy, authentication, integrity, and user autonomy. DLR offers land ownership proofs in digital form, granting users with control and authority over their land. DLR supports seamless and efficient land transactions by leveraging smart contracts and stablecoin payments, reducing the number of intermediaries. Finally, DLR supports an accurate definition of land limits with mapping and georeferencing techniques, following a GeoJSON norm to store property areas.

DLR is a decentralised application (DApp) that uses three blockchains: a smart contract platform (a permissionless blockchain), an SSI blockchain, and a cryptocurrency blockchain. The first allows users to register and execute transactions on their land, like buying or selling, leveraging smart contracts. The second allows users to have full control over their identity, the actions performed in their name and also over their land. The third is used for payments.

There is related work on blockchain-based land registry [1], [4], [5], [6], [7], [8], [9], which mostly considers land registry but not transactions. They also do not use SSI technologies. None of them provides all the characteristics of DLR. There are also related works on real estate transactions and rental, but not on land registration [10], [11], [12].

II. DLR System Design

This section presents the design of the DLR platform. First, two SSI concepts have to be introduced, both defined by the World Wide Web Consortium (W3C). Decentralised Identifiers (DIDs) are unique subject identifiers that enable verifiable and decentralised digital identities [13]. Verifiable Credentials (VCs) are digitally signed, tamper-evident digital

documents that can represent the same information that a physical credential represents [14].

A. Roles and Use Cases

1) Users: We use the term users to mean citizens, land owners, and private entities. To be able to use our platform, they need to have a decentralised identity, which is issued by the SSI Identity Agent owned by a government entity, interacting with the SSI ledger. This identity is represented by a VC and is used for registers, log-ins, and authentications on the platform. After this, users are able to register their land and execute transactions.

During the land registration process, users are required to provide information about their land, including its location, postal code, and GeoJSON mapping, which is a JSON file containing the coordinates of a polygon that defines the land boundaries. In addition, users need to submit a pdf document of the land registry record and, if they wish, to set the land for sale and define a price. This information is then automatically validated in a smart contract and, if successful, the SSI Land Agent issues a VC proving the ownership of the land.

Regarding the transaction process, users can buy or sell land using our platform. In this process, the owner of the land is required to submit the ownership VC related to the property in question. After this, if the smart contract's agreement terms are fulfilled, it is able to execute on its own and interact with a cryptocurrency network to send the agreed payment for the land from the buyer's to the seller's cryptocurrency wallet. At the same time, the SSI Land Agent revokes the ownership proof from the previous owner and issues an ownership proof to the new one; this way, the transaction is settled.

2) Government Entities: Government entities have a different role than normal users. One of their purposes is to maintain the land registry blockchain and execute consensus algorithms to decide whether or not registries and transactions are valid. In addition to this, these entities can assign government inspectors that can verify land registry information manually if needed. Government entities are also responsible for managing two SSI agents that issue VCs. The first is the identity agent, which is responsible for issuing decentralised identities to users containing their personal information. The second one is the land agent, which is responsible for issuing ownership proofs related to the user's land.

B. Design Options

During the design of the solution, several options were considered. In terms of blockchain type, a permissioned blockchain model was chosen to put authorised entities in control (government nodes, in our case). The technology chosen was Hyperledger Fabric, due to its modular architecture, flexible consensus mechanisms, scalability, privacy features, and strong community support.

The registry of users in our platform is based on an account creation using basic information like their name and fiscal number. As we wanted to employ a SSI solution, we decided that user identities would be tied to a previously issued identity VC, that needs to be stored in a SSI wallet. This means that after filling out a basic registry form, users need to connect their wallet to the platform. Then, a request is sent through the established connection and they are able to share this VC with our system, proving their identity. In this way, we defined the first SSI agent, the identity agent, which should be maintained by the government entity responsible for citizen IDs, and which is responsible for issuing these decentralised identities to users, containing their personal information and substituting physical IDs. Thanks to this decision, we enable users to register, login and authenticate themselves while being compliant with an SSI solution. The same logic applies to the login process, where authentication occurs through the wallet connection and the sharing of the identity VC. Moreover, when a land registry or transaction happens, the second SSI agent, the land agent, is able to issue a VC proving the ownership of the newly registered land. Overall, VCs are saved by the user and contain both their DID as well as the organisation's one and therefore, they can easily be validated and verified. As previously mentioned, Hyperledger Aries and Indy were the main SSI technologies studied in this work, due to their compatibility with Hyperledger Fabric and the possibility of issuing VCs. In this way, we decided to use them to implement the SSI layer of our system.

Land registry and transaction operations are implemented in smart contracts. In this way, we enable verifications before the contract executes, streamlining the whole process. Smart contracts also integrate payments in a simplified manner. In this case, using stablecoin cryptocurrencies to ensure quick and efficient transactions, reducing the time taken for settlements compared to traditional banking systems, fees, and also increasing the overall decentralisation of the system. Solutions like Stripe or PayPal were not considered, mainly because they are centralised and would require a lot of work with integrations and bureaucracy to be able to function with our system.

Regarding land mapping, we opted to do georeferencing only by submitting a file that contains land boundaries and complying with the GeoJSON geographic data format. In this way, we allow users to accurately define the limits of their land and to submit this information in a manageable format.

C. Architecture

The architecture of the solution is represented in Figure 1. Users interact with the system through the DLR Web App, which allows them to register or log in to the platform, register their land, and participate in transactions. As mentioned in the previous section, users need to have a decentralised identity to be able to use our platform. This identity is issued by the SSI Identity Agent owned by a government entity and interacts with the SSI ledger. This identity, represented by a VC, is used to both register and log in the platform, where the DLR Web Server requests the Identity Agent to validate the credential by checking the SSI ledger.

After the register and login processes, users can register their land and execute transactions by filling out predefined

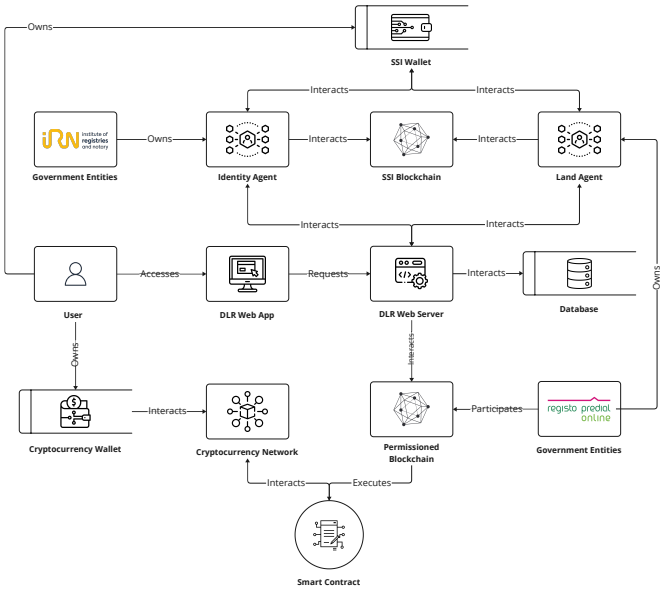

Fig. 1. DLR Architecture Data Flow Diagram

forms in the DLR Web App. If a user tries to execute a registry, the DLR Web Server interacts with the Permissioned Blockchain so that the respective smart contract is executed. In this case, the filled information is distributed between the participating government nodes (representing various government entities) who agree if whether or not the registration complies with the needed requirements. Each node should perform some automatic verifications, and, if needed, a government inspector is notified to manually verify additional information. If the registry process is successfully executed, the submitted information is stored in a manageable format inside the ledger. Subsequently, the DLR Web Server sends a request to the SSI Land Agent so that it issues a VC using its own DID to the user's SSI wallet, which contains the user's did. Therefore, the user now has an ownership proof that can be easily verified by anyone.

Regarding the use case of land transactions, the seller is required to submit his ownership VC related to the property in question before he is able to sell it. Buy requests are submitted by other users, and when the seller chooses the best offer, both users need to connect their cryptocurrency wallet, then the smart contract is executed. After this, if the terms of the smart contract's agreement are fulfilled, the payment is settled between the buyer and the seller. At the same time, the SSI Land Agent revokes the ownership proof from the previous owner and issues a new one to the new owner.

III. DLR Implementation

A. Permissioned Blockchain

1) Architecture: Our permissioned blockchain was implemented using the Hyperledger Fabric Kubernetes Operator (HLF Operator), which simplifies the setup of a production-ready Fabric networks and utilises Kubernetes orchestration. The HLF Operator allows the use of Custom Resource Definitions (CRDs) to configure the network's key components while also providing tools to manage them. The network consists of two organisations, representing government entities involved in the land registry process, each with its own Certificate Authority (CA) and two Peers. It also includes three Orderer nodes for fault tolerance and network availability. A channel, accessible only by these two organisations, ensures private communication for data sharing. The network also has a Gateway API SDK that supports interaction between external applications and the network.

2) Assets: In Hyperledger Fabric, an asset represents any tangible or intangible object or record that is tracked and managed on the blockchain. These assets are stored as key-value pairs, each of which is represented by a unique identifier (which is the key) and its associated properties (the values). We have two types of assets. The *Land Asset* represents the land itself and allows us to know its state, details, and ownership status. It contains properties describing the land itself, such as the location, GeoJSON and area, and properties regarding its status and ownership, such as the forSale Boolean, price, owner DID and credential exchange id. The owner DID is used in conjunction with the SSI Land Agent, allowing us to identify the identity of the owner, and the credential exchange id ties the Land Asset to the VC issued by the Land Agent. The *Land Transaction Asset* allows tracking the purchases and sales of land. It contains the ID of the land associated with the transaction. Besides this, it contains details about the seller and buyer, including their DIDs, cryptocurrency wallet addresses, and the price and status of the transaction (pending, accepted, rejected, completed). Besides these, both assets contain properties regarding their creation and update timestamps.

3) Smart Contracts: The smart contract used in our solution, created using Typescript, defines the business logic of the network. The smart contract functions are grouped into three types: general functions, which are private and retrieve asset information; land asset functions, which retrieve, create, and update land assets; and land transaction functions, which retrieve, create, update, and complete transactions. As some of the functions are complex and execute multiple validations and operations, we will describe them in more detail:

- CreateLandAsset: Starts by verifying if the land is already registered using the *assetExists* function. If not, it then checks if the GeoJSON data intersects with any existing land assets using the *DoesGeoJSONIntersect* function. If the GeoJSON data is valid, it creates a new land asset using the following properties: id, cred_ex_id, ownerDid, location, forSale, price, GeoJSON, and area and adds it to the ledger.
- UpdateLandAssetSaleStatus: Checks if the land exists and if the owner is the same as the one making the request. Then, if the forSale property is set to true, it updates the sale status and price of the land asset, only in case the price is valid. Otherwise, it sets the forSale property to false.
- UpdateLandAssetOwner: Is only accessible through the *CompleteTransactionAsset* function. It starts by checking if

the land exists, then it checks if the new owner is different from the current owner, and also if the current owner in fact owns the land. If successful, it updates the owner and the credential exchange id of the land asset.

- CreateTransactionAsset: Checks if the land exists and if it is for sale, then it creates a new transaction asset including the landId, seller and buyer DIDs, buyer cryptocurrency wallet address, price proposal and status to pending state. It then adds the transaction asset to the ledger.
- UpdateTransactionAssetStatus: Checks if the transaction exists and if the seller is the one making the request. If so, it updates the status of the transaction asset to either accepted or rejected based on the seller's request. In case it was accepted, it also adds their cryptocurrency wallet address to the asset.
- CompleteTransactionAsset: Checks if the transaction exists and if the buyer is the one making the request. If the request is valid, it updates the seller's cryptocurrency wallet address to verify the payment, the owner of the land asset, and the status of the transaction asset to completed, transferring the land ownership from the seller to the buyer.

B. Self-Sovereign Identity Layer

1) Architecture: As already mentioned, our SSI layer was built using Hyperledger Indy and Hyperledger Aries. Starting with Indy, which is a distributed ledger, we decided that it would not make sense to deploy an Indy network for our solution, as it would be too complex and costly, considering that there are open-source development networks available. Therefore, we decided to use the BC Gov's Von Network, which besides the network itself, includes a ledger browser to see the status of the nodes as well as to browse and search for ledger transactions. This network is being openly developed by the British Columbia government (BC Gov) which provides other useful sets of tools and libraries to build SSI solutions.[1] One of these libraries is called Traction and it was designed as an API first architecture built on top of the Hyperledger Aries Cloud Agent Python (ACA-Py) with the goal of streamlining the process of sending and receiving digital credentials for governments and organisations. Hyperledger Aries is a toolkit that allows for trusted peer-to-peer interactions based on decentralised identities and VCs. It includes several protocols and tools and supports identities across several blockchains. Aries serves as the client of an SSI system and allows us to issue, store, and present VCs through secure communication channels. However, ACA-Py is simply a cloud agent built on top of the Aries concepts, therefore serving the same purpose.

Traction is an interoperable tool that supports multi-tenancy, which is crucial in our case due to the two types of SSI agents in our system. It enables the deployment of digital trust services, leveraging verifiable data registries like Hyperledger Indy through ACA-Py. In this way, Traction offers a reliable and tamper-proof source of data on issuers and schemas, enabling secure sharing of digital credentials for organisations. Besides this, it offers an API first architecture allowing easy integration into existing applications, without the need to maintain and manage an instance of ACA-Py. Another key point is the existence of a Tenant UI, built on top of the API, allowing us to execute the key functions of the clients in a simple and user-friendly way, ideal for government entities managing the system. The most important components are the Traction Innkeeper UI, which allowed us to create our two tenants, the Identity Agent and the Land Agent, and the Traction API, which can be consulted through the Swagger UI and which allowed us to manage the agents and their interactions.

The last component of our SSI layer is the wallet used to store the VCs and manage the interactions with the two agents. In our case, we decided to use the BC Gov's SSI wallet, which is a mobile application that allows users to receive, store, and present digital credentials in a secure way.

2) Agents: Having created the agents using the Traction Innkeeper UI and receiving their respective wallet ID and private key, we were able to use the Traction Tenant UI to execute the initial configurations. This included registering the agents in the BC Gov's Von Network and receiving a ledger published public DID turning them into VC issuers. The second step to be able to issue VCs was to create the VC schema and credential definition and publish them in the ledger. The VC schema is basically its structure, representing the claims that the VC will contain, while the credential definition is the cryptographic material used to sign the VC. It links the Issuer's DID with the schema and contains the public key needed to verify the presentations of the credential, which can be done by anyone with the help of the ledger. We also created a Revocation Registry, linked to the credential definition, that allows the issuer to revoke credentials when necessary. This is useful when a land transaction happens and the old owner's VC needs to be revoked.

3) Verifiable Credentials: Following the process of defining and publishing the VC schema and the credential definition, we created the VCs that are used in our system, these being the Identity VC and the Land Ownership VC.

The Identity VC is issued by the Identity Agent and contains the user's name and fiscal number. Its purpose is to allow users to register and log in the system and, therefore, to prove their identity. For development purposes, we issued a few Identity VCs to some SSI mobile wallets, using the Traction Tenant UI, instead of creating a frontend dedicated to this single purpose, to be able to implement and test all the requirements of the system. This was considered not to be part of the scope of the paper, so in a real system, the Identity VC would be issued by the government entity responsible for the user's citizen IDs in a separate application.

The Land Ownership VC is issued by the Land Agent and contains the land ID, location, area, and GeoJSON of the land. Its purpose is simple, to allow users to truly have an ownership proof for the land they possess. This VC can be issued in two different scenarios, when the user first registers their

[1] https://www.lfdecentralizedtrust.org/blog/bc-digital-trust-leveraging-hyperledger-tools-for-digital-trust

land in our system, or after they complete a land purchase. Additionally, it can be revoked, in case the user sells the land and, lastly, it needs to be presented in case the user wants to prove ownership of their land. The schema of our Land Ownership VC was created with the goal of accurately representing any land. It is important to note that no predefined schemas for the land registry use case were found, and even if such schemas exist, they would have to both be suitable for our system and be registered in the Von Network Indy ledger, which is not the case.

4) Processes: Having understood how the SSI Agents work, we will now describe the main SSI related processes that occur in our system:

- User Registration and Login: Users can register and log in using the respective forms in the frontend. Starting with the registry process, the App sends a request to create a Single Use Connection link, which the User can scan with their SSI wallet to establish a secure connection with our system. With the connection in place, the App then sends a present proof request which the user can accept, sharing some of the information of their Identity VC, validating their identity in both the register and login scenarios.
- Land Registration: The user can register a new land by filling in the respective form in the frontend. The App then sends a issue credential request for the Land Ownership VC, containing the land details, which the user can accept, receiving their ownership proof for the land.
- Land Transaction: Land transactions occur after a user sends a buy request for a specific land. After the request is accepted by the seller, the App sends him a present proof request to make sure he is the owner of the land. Afterwards, the App sends an issue credential request with Land Ownership VC for the buyer, while also sending a revoke credential request for the seller, revoking their ownership of the land.

5) Traction API: The following are the key API functions we leveraged to help us manage the SSI Agents and their interactions with the user's SSI wallets:

- Authentication: Handles the process of retrieving a JWT token to be able to use the other API requests by posting the tenant's API key to the authentication endpoint */multi-tenancy/tenant/tenantId/token*.
- Creating Connection Invitations: Creates an invitation to establish a secure connection with another entity (in this case, the user's SSI Wallet) using the */connections/create-invitation* endpoint.
- Sending Presentation Requests: Issues a VC presentation request via the */present-proof-2.0/send-request* endpoint. The user receives this request in the SSI Wallet and accepts it, sharing the requested VC attributes. We use two types of requests: *identityPresentationRequest* to verify personal details (first name, last name, NIF); and *landPresentation-Request* to verify land credentials (location, area, geojson).
- Validating Presentations: Retrieves presentation proofs after the user accepts the presentation request via the */present-proof-2.0/records/id* endpoint. This validation is also divided into two functions to verify if the requested attributes (personal identity or land details) match the expected values.
- Issuing Credentials: Issues a Land Ownership VC by posting data to the */issue-credential-2.0/send-offer* endpoint. The process includes sending an offer to the user's SSI Wallet and issuing the credential if the user accepts.
- Revocation of Credentials: Revokes a credential using the */revocation/revoke* endpoint. We also check the revocation status via the */revocation/credential-record* endpoint.

C. Web Application

The DLR Web Application was built using Next.js, a modern React framework that enables the creation of server-side rendered applications. This framework allows us to create server actions, removing the need for a dedicated backend and API. Therefore, the Web App is composed by the Frontend and the Server Actions, while also interacting with two additional components. These include the Solana cryptocurrency network, for the land transaction payments, and a simple PostgreSQL database, to store the user's basic information as well as the connections to their SSI wallets. Regarding the land transaction payments, we chose the Solana network because it includes major tokens, such as USDC and EURC, which are Dolar and Euro stablecoins. Additionally, we chose it because of the low transaction fees and high performance, which are crucial for our system as we need to execute a high number of transactions in a short period of time. The actual implementation of the stablecoin payments was not included in the development due to their nonexistence in the Solana devnet. For the purpose of the demo we included payments with the native Solana token (SOL), which allowed us to test the payment process without incurring in real costs. Regarding the database, we created two tables: one for the users and another for the connections. The users table contains their name and fiscal number, as well as a representative ID and their DID (from the SSI wallet). This table is then related to the connections table, which contains the connection ID (between the SSI wallet and the Land Agent), an alias to identify the connection and the user ID.

1) Server Actions: The server actions are divided into two main types, Fabric Actions and SSI Actions, each one responsible for interacting with the respective component of the system. The Fabric Server actions use the Fabric Gateway API to interact with the underlying blockchain network, allowing our web application to submit transactions and query data directly from the smart contracts. For example, when a user registers a new land asset or submits a buy/sell request, server actions invoke the appropriate smart contract functions, simplifying the interaction with the blockchain while ensuring that all transactions are immutably recorded. An important point is that each smart contract function autonomously verifies the identity of the user who is making the request through their DID, ensuring that only authorised users can execute certain operations. This setup provides an efficient and secure mechanism for blockchain interaction, leveraging our Web

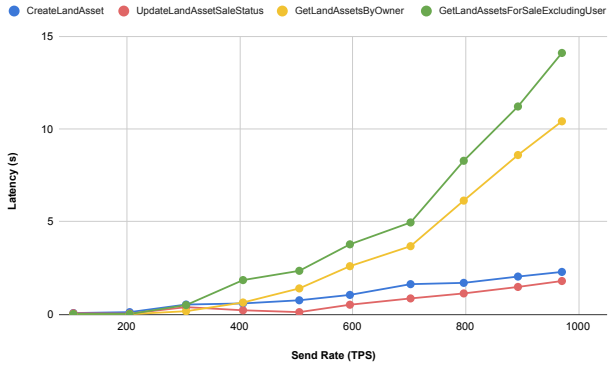

Fig. 2. Land Asset Functions Send Rate vs. Latency

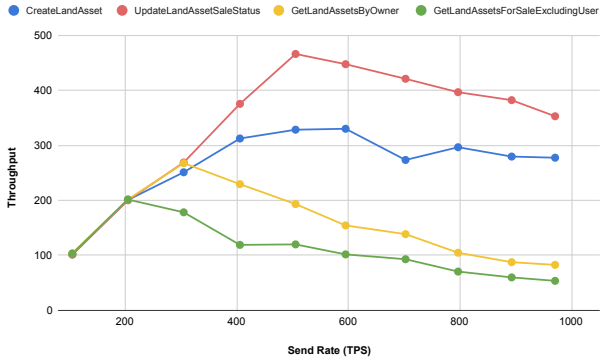

Fig. 3. Land Asset Functions Send Rate vs. Throughput

Server which acts as a trusted intermediary between the user and the Fabric network.

On the other hand, the SSI Server actions represent a tenant using the Traction API, allowing our app to manage VCs and secure connections with users' SSI Wallets. When a user registers or logs into the system, the server actions are responsible for creating connection invitations, sending presentation requests, and validating the user's credentials through the Traction API. For instance, when a user scans a QR code to establish a connection with the system, or when they verify their identity through their SSI wallet, the Traction API handles the validation of identity VCs. This ensures that the users' interactions with the web application remain private and fully decentralised, relying on the verifiable claims shared from their SSI wallet to validate transactions securely.

IV. Evaluation

This section summarizes the evaluation we did of the DLR platform, focusing on the performance of its blockchain layer. We used Hyperledger Caliper, a blockchain benchmarking tool designed to measure the performance of different blockchain networks. The benchmarks were executed on a MacBook Pro with a 10-core M1 Pro chip and 16GB of RAM. This machine provided sufficient computational resources for the execution of accurate measurements under varying loads, while maintaining a stable environment for both the Fabric network and the benchmarking tool. The benchmarking consisted of ten rounds, each for every smart contract function, with transactions per second controlled at a fixed rate to ensure uniform testing conditions. The number of requests per round began at 100 and scaled up to 1000, allowing us to measure how the system handles increasing transaction volumes. The workloads were carefully created to ensure that each smart contract function executed successfully under load, simulating real scenarios, where the execution parameters are valid and not supposed to throw execution errors.

Fig. 2 shows how latency increases with send rates for land asset functions. Fig. 3 presents how throughput varies with increasing send rates for the same land asset functions.

Acknowledgments. This work was financially supported by Project Blockchain.PT - Decentralize Portugal with Blockchain Agenda (Project no 51), WP6: Digital Assets Management, Call no 02/C05-i01.01/2022, funded by the Portuguese Recovery and Resilience Program (PPR), The Portuguese Republic and The European Union (EU) under the framework of Next Generation EU Program. This work was also supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).

References

[1] R. Khan, S. Ansari, S. Jain, and S. Sachdeva, "Blockchain based land registry system using Ethereum blockchain," *Journal of Xi'an University Architecture and Technology*, vol. 12, pp. 3640–3648, 2020.

[2] M. Themistocleous *et al.*, "Blockchain technology and land registry," *Cyprus Review*, vol. 30, no. 2, pp. 195–202, 2018.

[3] C. Allen, "The path to self-sovereign identity," https://www.lifewithalacrity.com/article/the-path-to-self-sovereign-identity/, 2016, accessed: 2024-12-12.

[4] R. Benbunan-Fich and A. Castellanos, "Digitization of land records: From paper to blockchain," in *ICIS 2018 Proceedings*, 2018, p. 15.

[5] S. Krishnapriya and G. Sarath, "Securing land registration using blockchain," *Procedia Computer Science*, vol. 171, pp. 1708–1715, 2020.

[6] A. Sahai and R. Pandey, "Smart contract definition for land registry in blockchain," in *IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*, 2020, pp. 230–235.

[7] M. Shuaib, S. M. Daud, S. Alam, and W. Z. Khan, "Blockchain-based framework for secure and reliable land registry system," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 5, pp. 2560–2571, 2020.

[8] A. S. Yadav and D. S. Kushwaha, "Digitization of land record through blockchain-based consensus algorithm," *IETE Technical Review*, vol. 39, no. 4, pp. 799–816, 2022.

[9] A. S. Yadav, N. Singh, and D. S. Kushwaha, "A scalable trust based consensus mechanism for secure and tamper free property transaction mechanism using DLT," *International Journal of System Assurance Engineering and Management*, vol. 13, no. 2, pp. 735–751, 2022.

[10] H. P. Wouda and R. Opdenakker, "Blockchain technology in commercial real estate transactions," *Journal of Property Investment & Finance*, vol. 37, no. 6, pp. 570–579, 2019.

[11] R. M. Garcia-Teruel, "Legal challenges and opportunities of blockchain technology in the real estate sector," *Journal of Property, Planning and Environmental Law*, vol. 12, no. 2, pp. 129–145, 2020.

[12] J. F. Santos, M. Correia, and T. R. Dias, "Blockchain-based rental documentation management with audit support," in *6th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, 2024.

[13] M. Sporny, A. Guy, M. Sabadello, and D. Reed, "Decentralized identifiers (DIDs) v1.0: Core architecture, data model, and representations," W3C, W3C Recommendation, Jul. 2022.

[14] G. Cohen, T. Thibodeau Jr, M. Jones, I. Herman, and M. Sporny, "Verifiable credentials data model v2.0," W3C, Candidate Recommendation, Sep. 2024.