frontiers | Frontiers in Blockchain

TYPE Original Research
PUBLISHED 15 June 2023
DOI 10.3389/fbloc.2023.1126978

OPEN ACCESS

EDITED BY
Adedoyin Ahmed Hussain,
Cyprus West University, Cyprus

REVIEWED BY
Stelvio Cimato,
Università degli studi di Milano, Italy
Tomasz Górski,
University of Gdansk, Poland

*CORRESPONDENCE
Nicu-Cosmin Ursache,
cos.ursache@gmail.com
Michael Sammeth,
micha@sammeth.net

RECEIVED 18 December 2022
ACCEPTED 24 May 2023
PUBLISHED 15 June 2023

CITATION
Sammeth M, Ursache N-C and Alboaie S (2023), OpenDSU: digital sovereignty in PharmaLedger.
Front. Blockchain 6:1126978.
doi: 10.3389/fbloc.2023.1126978

OpenDSU: digital sovereignty in PharmaLedger

Michael Sammeth[1,2]*, Nicu-Cosmin Ursache[3,4]* and Sînică Alboaie[4,5]

[1]Department of Anaesthesiology, Intensive Care, Emergency and Pain Medicine, University Hospital Würzburg, Würzburg, Germany, [2]Department of Applied Sciences and Health, Coburg University, Coburg, Germany, [3]Faculty of Computer Science, Alexandru Ioan Cuza University, Iași, Romania, [4]RomSoft SRL, Iași, Romania, [5]Axiologic Research SRL, Iași, Romania

Introduction: Distributed ledger networks, chiefly those based on blockchain technologies, currently are heralding a next-generation of computer systems that aims to suit modern users' demands. Over the recent years, several technologies for blockchains, off-chaining strategies, as well as decentralised and respectively self-sovereign identity systems have shot up so fast that standardisation of the protocols is lagging behind, severely hampering the interoperability of different approaches. Moreover, most of the currently available solutions for distributed ledgers focus on either home users or enterprise use case scenarios, failing to provide integrative solutions addressing the needs of both.

Methods: Herein, we introduce the OpenDSU platform that allows to interoperate generic blockchain technologies, organised—and possibly cascaded in a hierarchical fashion—in domains. To achieve this flexibility, we seamlessly integrated a set of well conceived components that orchestrate off-chain data and provide granularly resolved and cryptographically secure access levels, intrinsically nested with sovereign identities across the different domains. The source code and extensive documentation of all OpenDSU components described herein are publicly available under the MIT open-source licence at https://opendsu.com.

Results: Employing our platform to PharmaLedger, an inter-European network for the standardisation of data handling in the pharmaceutical industry and in healthcare, we demonstrate that OpenDSU can cope with generic demands of heterogeneous use cases in both, performance and handling substantially different business policies.

Discussion: Importantly, whereas available solutions commonly require a pre-defined and fixed set of components, no such vendor lock-in restrictions on the blockchain technology or identity system exist in OpenDSU, making systems built on it flexibly adaptable to new standards evolving in the future.

KEYWORDS

opendsu, blockchain interoperability, off-chain data, blockchain anchoring, self-sovereign identities, pharmaledger, blockchain in healthcare, blockchain in the pharmaceutical industry

1 Introduction

In a world that is becoming more digital than ever before, new technologies are emerging to satisfy the needs of modern users to connect with each other. Recently, the development of distributed ledger technologies (DLTs) has fueled a plethora of novel concepts regarding the exchange of information, data and other digital assets (Section 3).

Technically, most of the successful DLT approaches rely on blockchains, which redundantly store and process user transactions in a distributed network. In the remainder of this section, we briefly sketch challenges and chances of these decentralised DLT systems. Especially over the last decade, services for every aspect of life have increasingly become accessible through internet platforms. Already during the commercialisation of the world wide web, infrastructure gradually shifted from "web1", where each user as an equivalent node could produce and/or consume content, to today's "web2" structure (Dixon, 2018) where centralised platforms operated by a small group of "Big Tech" companies administrate services required by home users and/or smaller companies (Marlinspike, 2022). Recently, the concept of a next-generation "web3" architecture promoting the return to decentralisation (Edelman, 2021) is increasingly drawing attention. Although concrete visions of web3 are still varying, a common key concept is that both the ownership and the control of internet services should both be decentralised again. Decentralised ownership of exchangeable assets is usually broken down by tokens (Kubach et al., 2020), and non-fungible tokens (NFTs) have been conceived to represent digital counterparts of unique assets such as artefacts, credentials, governance rights, access passes, etc (Alblooshi et al., 2018). The decentralised control of services is achieved through internet platforms that democratise and streamline user interactions based on DLT networks instead of relying on a centralised database.

Later web3 platforms (Blockchain Council, 2022) are commonly known as decentralised applications (dApps), e.g., dApps for Decentralised Finance (DeFi) allowing users to exchange currency without any involvement by a bank or government (Schär, 2021). On the top of such decentralised architecture, web3 supporters further see a novel form of organisations substituting the traditional coordination patterns of corporate business in the long run: these so-called Decentralised Autonomous Organisations (DAOs) are internet-based companies that are collectively owned and controlled by their respective members (Hassan and Filippi, 2021). Blockchains crystallise best technologies to achieve this far-reaching goal of interoperability, since they not only allow for fast confirmation but also for transparent and cryptographically secure transactions. However, today's platforms are still falling for the web2 centralisation traps, as most of the web3-based concepts currently remain visionary and are lacking concrete implementations of the necessary tools.

To this end, the main contribution of our present work consists in developing a novel approach to build blockchain platforms, which we call "OpenDSU" and which we make publicly available under the MIT Open Source License. One of the unique features in our OpenDSU approach is the cryptographic key management, which offers novel types of managing cryptographic keys and novel ways of organising data sharing between digital wallets. The remainder of the paper is organised as follows: Section 2 summarises our investigation of essential requirements for a blockchain platform, and Section 3 subsumes our research on the viability of existing solutions. In Section 4, we introduce the different components of our novel platform, before we describe in Section 5 our insights from employing OpenDSU to challenges of the recently initiated PharmaLedger Project. Our general findings and possible extensions are discussed in Section 6. Finally, Section 7 concludes the paper with aspects of limitation and some future directions.

2 Problem definition

2.1 Sovereign identity systems

To access services on internet platforms, users need to transfer their own identity to the digital world. To enable trustful interactions between digital identities, the organisation-centric ID management authenticates users based on passwords that are controlled separately by each data provider in the form of a traditional database, resulting in obvious privacy concerns. Later-on, in the progress of web2, the federated identity systems have emerged, allowing users to verify their digital IDs by password-based credentials also with third parties, without exposing personal information to each of them. However, whilst reducing the number of data providers with access to private user data, concerns about potential data leakage and abuse by malicious identity providers can obviously also not be refused in federated systems.

Therefore modern DLT systems are moving towards more user-centric approaches to identity control, like Decentralised Identifiers (DIDs) in general allowing for Self-Sovereign Identities (SSIs) in particular (Kubach et al., 2020). Operating without centralised registries, identity providers, and certificate authorities by design, DIDs can be used to represent people but also objects (Alblooshi et al., 2018), datasets (Barclay et al., 2020), and many more. The World Wide Web Consortium (W3C) recently released a DID recommendation based on ten principles (Sporny et al., 2021b) as well as a recommendation on Verifiable Credentials (VCs) to promote verification processes and by this to "bind" a certain user to some data or transaction (Sporny et al., 2021a).

In this light, VCs are associated with digital identities in the same way physical credentials are tied to traditional identities. However, VCs are designed for web environments, where it is more difficult to verify or validate the information because plain digital patterns are more easily tampered with than their physical-biological counterparts. VCs therefore require to be cryptographically secure, privacy-respecting, and machine-verifiable. Driven by motivations to preserve a maximum of privacy, VCs are postulated to reveal a minimal amount of personal information exposed by the users–a maxim for which the term Zero-Knowledge Proofs (ZKPs) has been coined (Goldwasser et al., 1989). Employing ZKPs, the issues in an identity system are shifting from the former concerns about potentially malicious data controllers to new concerns about potentially malicious users, bringing up the question of how to create trust in a trustless digital environment. Therefore, a major challenge of introducing SSIs in DLT systems remains the fine tuning of the balance between the opposingly seeming requirements of privacy and trust.

2.2 Blockchains and scalability

As compared to distributed systems where each site has its own administrative control over the assets (e.g., grid computing), the downside of decentralised blockchains is that data on the blockchain

TABLE 1 Use Case Patterns. Use cases from permissioned networks are grouped into four distinct collaboration categories ("use case patterns"), according to characteristics of the stakeholder(s) involved, their privacy requirements and possibilities to prevent spamming attacks.

	Pattern #1	Pattern #2	Pattern #3	Pattern #4
Use Case Pattern	Single Data Provider	Decentralised Choreographies	Trustless Choreographies	Global/Public Networks
Example Use Cases	NFTs and Data Anchoring	Business Processes	DAOs	Crypto currencies, DeFi systems
Stakeholders	Single entity sharing data	Business partners updating data	A large group of independent peers	A network of independent parties
Privacy	Encryption, P2P messaging	Encryption, P2P messaging	ZKP, Encryption, Layer 2 solutions	ZKP
DoS/Spam Prevention	—	Legal contracts	ZKP, Transaction fees	ZKP, Transaction fees

(i.e., "on-chain" data) and corresponding computations are replicated redundantly by all the nodes in the network. By this, computationally intensive transactions in combination with the performance of the least performant node can slow down the blockchain network, hampering the throughput and also the scalability of the entire system. According to the classification of (Eberhardt and Tai, 2017), such "computationally intensive" transactions consist of transactions that include costly computations of deferred results, transactions with accumulating data, or transactions with heavy data files.

Naive attempts to circumvent such performance issues by keeping large files separately on premises beyond the blockchain ("no-chain" data) result in a loss of data integrity and auditability, as well as in issues with the protection particularly of sensitive data. Therefore an increasingly explored approach consists in computing hash codes that are kept on-chain before relocating heavy or sensitive files to systems outside the blockchain ("off-chain" data).

However, importing off-chain data to/from external premises also raises novel challenges: first, the volume of off-chain data often grows manifold faster than on-chain data, e.g., >40-fold in an empirical study on a 5-node Hyperledger system (Corporation, 2018); second, a centralised off-chain storage re-introduces previously discussed problems of data control, whereas a decentralised off-chain storages questions about security, availability and delivery of the off-chain data; third, off-chain data needs to be validated upon re-import, e.g., by leveraging hashing techniques, imposing additional overhead to the system. In a nutshell, current key challenges for off-chain storage strategies comprise data access regulation, storage security, cryptography, overall performance, and system scalability.

2.3 Use case patterns

Ledgers are employed to model a variety of use cases from different business sectors, such as financial accounting, logistics, healthcare, etc. These use cases inherently differ in the number and degree of mutual trust of the users interacting with each other, and consequently also in the challenges for implementing them. Based on our research on best practices for programming ledger solutions, we classify use cases in one of four abstract interaction scenarios we call use case patterns (Table 1):

*Pattern #1 (Single Data Provider) d*escribes use cases, where a single entity is sharing its data with partners. In a typical use case of this category, a–potentially big–company ("producer") is providing controlled read-only access to its NFTs for their partners ("consumers"). Although scenarios along these lines could be solved already by employing traditional databases, blockchain solutions are today often preferred due to the inherent advantages of decentralisation in offering data integrity and audibility. In fact, most enterprise blockchain use cases classify either directly as *Pattern #1* or can be redesigned in a way that every data item has a single controller. Privacy is usually assured by direct peer-to-peer (P2P) messaging and encryption. Denial-of-Service (DoS) attacks, such as adversary spamming, and Replay Attacks are not relevant to *Pattern #1* use cases, since the only meaningful attacks could originate from the data controller itself.

Pattern #2 (*Decentralised Choreographies*) comprises use cases within a–usually small–group of business partners who are interchanging their respective data, such as for instance in the banking sector or in multi-centric clinical studies. Since the partners involved in the business processes mutually trust each other, privacy can still efficiently be preserved by encryption and P2P messaging. As in Category *Pattern #1* use cases, DoS attacks are of subordinate relevance because all transaction partners are bound with each other by legal contracts. This pattern comes as a good solution for the corner cases where *Pattern #1* does not suffice. For instance, modelling the data stats of a business process between multiple parties represent a very general use case for *Pattern #2.*

Pattern #3 (*Trustless Choreographies*) describes use cases amongst parties in a–usually larger–group, usually spawning (sub-)clusters of cooperating partners or otherwise not one-to-one connected peers. The fact that the pattern includes several stakeholders hampers straightforward extensions of solutions for use cases from *Pattern #1* and *Pattern #2* in their confidentiality protection, and makes them vulnerable to spam attacks. Therefore, technologies like ZKPs should be deployed (Section 1). A typical use case of pattern *#3* could be constituted by a network of public DAOs that have smart contracts but are independent of other DAOs, respectively, they do not need to sync in real-time with the status of the whole network. The natural segregation of the users in the entire network into largely independently operating clusters enables various performance optimisations, that are explored, e.g., by the side-, child- and off-chaining approaches in Layer 2 solutions (Section 3.3).

Pattern #4 (*Global/Public Networks*) subsumes all use cases in which anonymous parties interact with each other through transactions in a decentralised ledger network. Classical examples are the popular cryptocurrency blockchains (e.g., Bitcoin), DeFis, or other networks with transactions of valuable things. In such

scenarios, the whole network has the interest to ensure correctness. As for use cases of *Pattern #3*, ZKPs are required to verify the identity of anonymous entities in such public networks. The subtle difference between the *Pattern #3* and *Pattern #4* is that in pattern *#3* some off-chain data integrity can be tolerated without serious risks, and on-chain data integrity can be minimised but not entirely removed.

In our work, we focus on use cases that commonly incur in the context of enterprise *consortia*, which according to our observations predominantly match the characteristics described above as Use Case *Pattern #1* or *Pattern #2* (Section 5.3). The essential pre-requirements of a consortium network between different companies, possibly based in different countries, can be subsumed by.

(i) each partner/company requires sovereignty over the own domain,
(ii) each partner/company may act as a data producer or consumer at the same time,
(iii) subsets of partners/companies need to interchange some of their data in a controlled and secure manner.

Observation *1*) points out that a for an enterprise consortium successful blockchain platform is required to provide a SSI-capable DID framework (Section 2.1). Furthermore, observation *2*) implies that each partner can produce–potentially considerable–amounts of data, which need to be stored off-chain in order to not compromise the performance and scalability of the blockchain network (Section 2.2). However, since substantially different business policies might apply to data produced by the one or the other partner in the consortium, these data often cannot be stored in a common place but need to reside at premisses controlled by the corresponding data owner. Finally, the successful realisation of pre-requirement *3*) depends on a generalised data sharing strategy that allows the permission-controlled, blockchain-validated and cryptographically secure exchange of distributed off-chain data between the limited subnetwork of consortium partners participating in a transaction (cf. *Pattern #1* and *Pattern #2* in Section 2.3).

3 Related work

A vast variety of DLTs has been developed over the recent years, however, comparatively little effort has been spent on the systematic classification, evaluation and standardisation of different approaches. Consequently, today's landscape of ledger solutions is marked by a high degree of diversification in their concept of smart contracts and communication patterns, originating also from the heterogenous challenges of the use cases for which they were designed. Focusing on permissioned ledger technologies, we subsume in the remainder of this section some popular DLTs that are related to our work (Pritzker et al., 2021c).

3.1 UTXO models in public ledgers

Going back to the famous Bitcoin system (Nakamoto, 2018), global ledgers have been conceived principally to address the use cases described as *Pattern #3* (permissioned/private) and *Pattern #4* (unpermissioned/public) in Table 1. A common attribute of global ledgers is their use of a global transaction model called UTXO (unspent transaction output). Under this model, UTXO represents the current state of a global database, from which transactions can "spend" or "consume" assets as inputs, and corresponding outputs then are incorporated in the updated UTXO database. In contrast to ledgers employing account models (Section 3.2), smart contracts in UTXO systems are essential for guaranteeing global data integrity, checking the validity of every submitted transaction with respect to its correctness, authorisation and uniqueness.

3.1.1 Corda: smart contracts and P2P messaging

Developed mainly by the company R3 with other partners involved, Corda [1] is an open-source project that (inter-)operates permissioned networks with Smart Contracts executed in a JVM-compatible language (i.e., Java or Kotlin). Focusing originally on use cases of the financial industry, a Corda ledger supports smart contracts that may contain an associated legal prose (i.e., Ricardian Contracts (Grigg, 2004)), providing a root for the legitimacy of the corresponding code. Corda has been designed from the ground up to support a global network of smaller business networks (*Pattern #3* in Table 1) and recently also offers a DID method implementing the W3C standards (Kirtani et al., 2020).

As a key differentiator to other DLT approaches, the existence of a global state for the entire blockchain is optional (usually not implemented). Corda transactions still are cryptographically linked to the transactions they depend on, but communicated P2P exclusively to involved nodes for processing. Transactions thus are approved "immediately", increasing the throughput of the entire system, but nodes can "see" exclusively those transactions in which they are participating and are consequently limited in their ability to validate a transaction involving user(s) from another subnet. In such cases, additional overhead needs to be spent by inquiries through a specialised "notary" node. Corda networks fit well use cases in environments of highly regulated clusters, like services of the financial industry, where the global network can be broken down into multiple sub-networks.

3.1.2 MultiChain: permissions and predefined rules

Forking off the public Bitcoin blockchain, the MultiChain[2] approach has been developed as a private, permissioned network that–as suggested by its name–allows to interoperate multiple parallel, but not hierarchical, instances of the blockchain with cross-chain applications (Greenspan, 2015). Technically, the MultiChain approach addresses two general bottlenecks of blockchains: privacy is enabled by "streams" that only are accessible to users with the corresponding privileges, and scalability is improved by an integrated off-chaining strategy (Section 2.2). Only the hashes of heavy data are stored on-chain whereas the data itself resides outside the chain, either in form of a *1*) centralised repository, or *2*) handled by a P2P file sharing system, or *3*) stored in local repositories that are managed through MultiChain itself.

1 https://www.corda.net
2 https://www.multichain.com

A central difference of MultiChain lies in the implementation of rules that control transactions: smart contracts are predefined rather than established by the users, which merely may select for each use case a certain level of validation stringency by disabling some of the predefined rules through so-called "stream filters". The stream concept in MultiChain ledgers thus allows users with corresponding permissions to set up additional scopes for groups in addition to the "root stream" of a blockchain (*Pattern #3* in Table 1).

3.2 Account models for enterprise solutions

The UTXO paradigm (Section 3.1) is well suited for global ledgers with similar transaction types, its limited set of smart contracts hampers a flexible implementation of generic business processes, where transactions are largely independent of each other and do not affect many pieces of information at the global scale. Therefore, enterprise solutions predominantly employ "account" models that bind smart contracts to the data they are controlling within the ledger. By this, the scope of data that can be altered by a piece of chain code, and *vice versa* also the code that is able to modify certain data in the ledger, is strictly defined. Corresponding ledgers are modular systems of atomary databases where data is exchanged in the form of messages to and from smart contracts controlled through the "accounts" of the users. Such messaging often provides a more natural approach than UTXO to model use cases of *Pattern #3*.

3.2.1 Quorum: hybrid network with public and private transactions

Developed originally by the investment bank JPMorgan and later-on acquired by the ConsenSys software company, Quorum[3] is an open-source, enterprise-focused, permissioned blockchain that has been developed off the official Ethereum protocol[4]. In contrast to Ethereum, however, Quorum exclusively admits participants to the network who have been pre-approved by a designated authority, the so-called "consortium". This avoids potential threats of catastrophic failure or security breach, and at the same time enables the exchange of private P2P-transactions in addition to the public messaging inherited from Ethereum. Quorum therefore is considered a "hybrid" architecture in the language of (Ismail and Materwala, 2019).

With ZKP enabled technologies available, Quorum is in principle suited to model use cases of any of the four patterns described in Table 1. However, the constellation P2P messaging is cryptographically not perfect and under certain circumstances (i.e., with pre-knowledge of the network mapping) nodes can gather some basic activity data about private transactions that they are not involved in (Tanner and Khan, 2021). Moreover, Quorum also does not natively provide off-chaining strategies for the private states of the nodes, which can lead to scalability issues when the number of data exchanged privately grows big.

3 https://consensys.net/quorum/

4 https://ethereum.org/en/

3.2.2 Hyperledger Fabric: a multi-ledger network

Also Hyperledger Fabric[5] (simplified denoted "Fabric") is an open-source, permissioned, private blockchain system, which has originally been initiated by Digital Asset and IBM[6] but now evolved as a collaborative cross-industry venture hosted by The Linux Foundation (Androulaki et al., 2018). To enable privacy, the Fabric architecture divides the network into "channels" and the data into "private data collections". This two-level privacy mechanism provides flexibility in modelling different business policies, but also impacts negatively on the performance, since creating a channel within a channel requires intense computing (Thakkar et al., 2018).

Another key difference of Fabric is the "endorsement" approach, which provides a high level of flexibility for the chaincode and still guarantees determinism: before committing a message to some chaincode a peer is required first to send the corresponding message to some endorsing peers for independent execution, to receive their opinion on the safety in form of an "endorsement". However, as all the endorsing and committing peers perform the encryption/decryption operations, the endorsement policy is also costly in computations. In a nutshell, Fabric multi-ledger-based architecture can be flexibly used in use cases involving several collaborating organisations, particularly as outlined by our *Pattern #2* and *Pattern #3* (Table 1).

3.3 Layer 2: interoperable blockchains

Layer 2 refers to a secondary system that sits on top of a blockchain network to relieve some of the workload employing side-chaining, child chaining, or off-chaining approaches. One first step in this direction was the "side-chaining" approach originally proposed by the Elements platform[7] to enhance performance and privacy of the public Bitcoin blockchain (Nakamoto, 2018). Elements side-chaining employs special nodes called "Watchmen", which move ("peg in") multi-signature transactions for verification from the main blockchain to a separate blockchain, and subsequently back to the main chain ("peg out").

3.3.1 OpenChain: side-chaining with anchors

The OpenChain platform[8] developed by CoinPrism later-on leveraged such Watchmen concept for blockchain interoperability employing the Lisk platform[9]. Through Watchmen OpenChain "freezes" cumulative hashes of transactions created in (possibly private "close-loop") side chains to a public main blockchain (i.e., a Bitcoin system) in a process called "anchoring". This allows OpenChain to validate side-chain transactions without adopting the concept of a block, and the responsibility of immutability is delegated through anchors to the public blockchain. However, OpenChain's "side-chains" in fact do not

5 https://www.hyperledger.org/use/fabric

6 https://www.ibm.com/topics/hyperledger

7 https://elementsproject.org

8 https://www.openchainproject.org

9 https://lisk.com

employ blockchains but structure transaction data that can be interlinked employing hash anchors from the main chain.

3.3.2 Bitcoin lightning: cryptocurrency side-chains

The Lightning Network[10] is a system that is built on top of a public blockchain to facilitate fast peer-to-peer transactions. The most popular application of the Lightning Network is in combination with a Bitcoin blockchain, but it constitutes a separate solution and also other cryptocurrencies such as Litecoin have integrated it. Separated from the Bitcoin network, the Lightning Network operates with its own nodes spawning "private channel" networks that can be seen as mini-ledgers for fast P2P transactions that are not dependent on building blocks. Special transactions are necessary on the main blockchain to initiate or end a private channel, involving locking of the assets that can be spent in the P2P side-chain. Moreover, in the Lightning Network the side-chains must be of the same nature as the main blockchain.

3.3.3 Ethereum plasma: hierarchical child chains

Plasma chains[11] in Ethereum are separate child blockchains anchored to the main Ethereum blockchain by Merkle trees. In principle, each Plasma child chain can be interpreted as a customisable smart contract that is designed to serve particular needs. The communication between the child chains and the root chain is secured by so-called "fraud proofs", which delegate to the root chain the responsibility of keeping the network secure and of propagating malicious actors across child chains. The tree structure allows the anchoring of child chains also cascaded in hierarchies, but they all are required to run the same blockchain technology in order to guarantee compatibility with the main Ethereum blockchain.

Beyond these general blockchain approaches, recently some optimised protocols for blockchain interoperability in special contexts have been proposed, e.g., for leveraging the interoperability of supply chains (Viriyasitavat et al., 2022) and within the structural hierarchy of controlling organisations in a smart city (Rahman et al., 2022). However, since the focus of these developments lies in the interoperability of defined blockchains amongst *a priori* known partners in particular use cases, these approaches are difficult to generalise to arbitrary use cases that may even require to operate on different blockchain technologies. Therefore, a general solution to exchange data of substantially varying structures and policy constraints across use cases, blockchain technologies and self-sovereign domains remains challenging in consortia networks.

4 Methods: the OpenDSU platform

Although recent Layer 2 approaches allow operating multiple blockchains of the same or very similar DLT to distribute the workload, mainly of financial transactions in cryptocurrency networks, our elaborations in Section 2 demonstrate that currently there is no platform to orchestrate multiple, arbitrarily

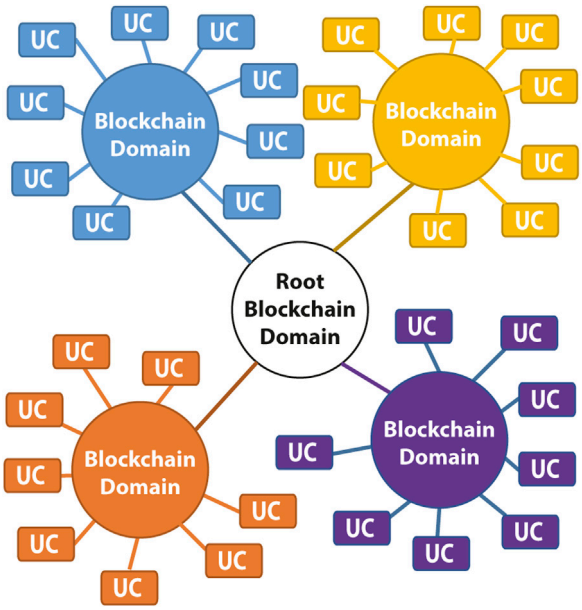

FIGURE 1
Blockchain Domains. The Open-DSU approach orchestrates a multi ledger network across different blockchain domains, each one hosting possibly multiple use cases (UCs).

different DLTs in a single enterprise environment. Going back to the PrivateSky Project[12], we therefore sought to develop a platform we call "OpenDSU", which is to fill in this gap. Since then, our OpenDSU SDK[13] matured and improved by employing it in research and commercial projects - such as the currently ongoing PharmaLedger Initiative[14] (Section 5). The vision of OpenDSU is to provide a framework for combining blockchains and other DLTs, existing ones such as the ones developed in the future, without any pre-requirement on their architecture and without compromising performance and privacy (Figure 1).

Some of the key contributions by OpenDSU are

- *Digital sovereignty*: the OpenDSU concept was designed from the ground up to allow nodes with SSIs from different so-called blockchain domains, intrinsically linked to a finely granular permissioning system (Section 4.4).
- *Enterprise smart contracts*: in the spirit of the smart contract concept outlined in Section 3.2, OpenDSU binds chain code to the data it controls, in containers called DSUs ("data sharing units", Section 4.2).
- *Off-chaining*: based on these DSUs, OpenDSU introduces a universal off-chaining strategy (Section 2.2) that allows storing of so-called near-chain data across different blockchain domains as "bricks" in "bricks storages" (Section 4.3).

10 https://lightning.network
11 https://ethereum.org/en/developers/docs/scaling/plasma
12 https://profs.info.uaic.ro/~ads/PrivateSkyEn/
13 https://opendsu.com
14 https://pharmaledger.eu

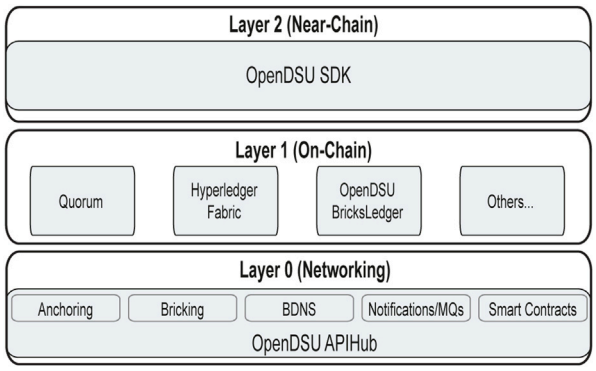

FIGURE 2
Layers of an OpenDSU platform. Bottom: Layer 0 is the networking layer that provides the interconnectivity between the replicas of the blockchain domain(s). Centre: Layer 1, the "on-chain" layer, provides possibly various ledger technologies with their respective consensus-building methods. Top: Layer 2 consists mainly of the OpenDSU SDK library and manages the near-chain processing.

- *Multi-chaining*: starting off from the idea "no size fits all", OpenDSU leverages anchoring techniques (Section 3.3) to bind near-chain data to possibly hierarchic ledger network(s), shaping a technology-agnostic, multi-chain platform (Section 4.3).
- *Off-the-shelf platform*: the source code of all OpenDSU components is transparently available and can be employed "out of the box", without any vendor lock-in restrictions.

In the remainder of this section, we present in more detail the key components of our OpenDSU platform (Alboaie et al., 2022i).

4.1 Overview and architecture of the OpenDSU platform

Figure 2 sketches the logical layout of any OpenDSU platform. From a technical point of view, all software components can be segregated into three structured layers, i.e., the network (layer 0), the on-chain (layer 1), and the "near"-chain layer (layer 2).

4.1.1 Layer 1: the ledger network(s)

The centre panel of Figure 2 shows the actual DLT networks, which we consider the OpenDSU "Layer 1", each one running several replicas. In OpenDSU we have support for Quorum (Ethereum) blockchains (The Federal Reserve Bank of Boston, 2019), Hyperledger Fabric blockchains (experimental) (The Linux Foundation, 2020; Frankenfield, 2021), or instances of our OpenDSU BricksLedger (Section 4.3) network (Thakkar et al., 2018; Alboaie et al., 2022g,h). Further DLTs may be added in the future, requiring merely a correspondingly implemented and properly configured *adapter*.

In our OpenDSU paradigm, the primary purpose of blockchain(s) is to work as notarization mechanisms for DSUs, allowing the integration of different blockchains that may provide, for instance, different security models and/or scaling capabilities. To this end, OpenDSU has been conceived from the start to be agnostic towards the employed blockchains. In order to enable the interoperability of heterogeneous and not *a priori* fixed DLTs, OpenDSU provides an independent naming system we call BDNS (Blockchain Domain Naming System) (Alboaie et al., 2022b). BDNS is essentially a discovery mechanism that translates numerical addresses needed for locating and identifying computer services and devices of the underlying network protocols to more easily memorisable domain names in the form of "subdomain.rootdomain.topdomain". In this light, BDNS is a service for blockchain bootstrapping by trusted configurations, similar to DNS, DPKI, and other forms of verifiable mapping, which however in contrast to the latter aims to be a secure, hierarchical, decentralised and self-sovereign naming system (Alboaie et al., 2022b). BDNS enables smart contracts on hierarchical blockchain domains, agnostic to the respective DLT employed (Alboaie et al., 2019).

4.1.2 Layer 2: the OpenDSU SDK

Layer 2 (Figure 2, top panel) consists of the OpenDSU SDK, a collection of high-level functions associated with the "blockchain domains" endpoints, specified by an hierarchical path in the form of "component > blockchainDomain > action > parameters". Whereas the traditional nomenclature for blockchain systems usually distinguishes merely data stored in the ledger (on-chain) from off-chain data that can be controlled independently, we further discriminate three classes of the latter.

(i) near-chain data is wrapped into so-called DSU containers (Section 4.2), which can be exported and validately reimported to and from external premises employing "bricking" and anchoring techniques (Section 4.3).
(ii) far-chain data in OpenDSU is exported from the ledger without anchoring to a blockchain, e.g., by employing databases (Pritzker et al., 2021a).
(iii) no-chain data is used by the OpenDSU platform, but not employed by the DLTs in Layer 1 at any time. This refined off-chain policy allows us to differentially outsource from the blockchain such shared data, which needs consistently to be accessed by different stakeholders (near-chain) as well as sensitive data that often follows business-specific policies (far-chain).

4.1.3 Layer 0: the APIHub

The networking layer or Layer 0 (Figure 2, bottom panel) comprises different components (e.g., BDNS, bricking, anchoring, etc.) running in the so-called APIHub, the backend of our OpenDSU platform. The APIHub acts as a server that offers–typically to wallets–different functionalities provided by the OpenDSU SDK (Layer 2). According to their specificity for a DLT, we further distinguish adapter-components relying on specifically implemented blockchain adapters (e.g., BDNS) from adapter-independent components (e.g., bricking). The OpenDSU APIHub is extensible, i.e., new components may be added in custom implementations. In our present OpenDSU implementation, all

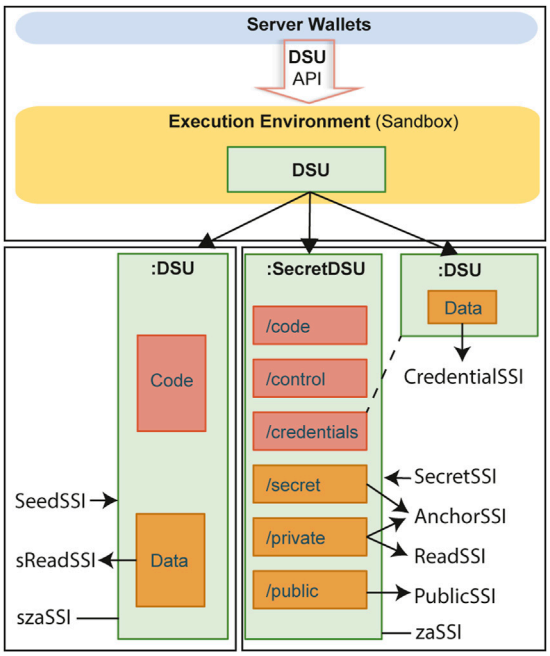

FIGURE 3
Overview of the DSU core concept. Lower-Right: Lower-Left: SecretDSUs employ the SecretSSI family of KeySSIs with the derivation hierarchy: SecretSSI (DSU ownership) > AnchorSSI (writing) > ReadSSI (reading) > PublicSSI (basic DSU access) > zaSSI (reference without access).

APIHub computations are being executed in a single/main thread, but our ongoing efforts comprise the development of a multi-threaded APIHub to increase performance of the OpenDSU platform.

4.2 Sharing data with DSUs

4.2.1 Simple DSUs

In OpenDSU–as suggested by the name–data sharing units (DSUs) are central components, which we designed from the start as near-chain containers (Alboaie et al., 2022j). When exported to external storages (Section 4.3), DSU contents are encrypted and tokenized to provide confidentiality and privacy. They can only be decrypted and re-assembled to a DSU after a suitable access key (a "KeySSI", Section 4.3) has been resolved by a server-, cloud- or edge-based wallet. Once the so-called KeyResolver resolves a KeySSI to a certain DSU, the corresponding near-chain data–provided the necessary access level–is decrypted and loaded in the execution environment.

Figure 3 (upper panel) shows that, after assembly, a DSU instance becomes available in the corresponding execution environment, usually a sandboxed container (Sabt et al., 2015). A DSU spawns a micro file system that also can be interpreted as a key-value micro-database, with the path of each file representing the key pointing to the file's content. DSUs can contain both, near-chain data and also chain-code (i.e., smart contracts), which is not visible on-chain and therefore can also be considered a form of "secret" smart contracts.

Figure 3 (lower left panel) outlines the finely granular access control to DSUs based on so-called families of keySSIs, created from one another by "derivation" (Section 4.3). For instance, a KeySSI family popularly employed with DSUs that are shared only amongst a rather limited set of users starts off with the generation of a private key that provides full control (i.e., ownership) of the DSU, the SeedSSI. From this, a hashed version called "sReadSSI" (*seed read SSI*) then is "derived" to be shared amongst the group of users who encrypt and decrypt the collective data. Further derivation produces a "szaSSI" (*seed zero access*), a public key that provides no access to the DSU itself but exclusively to a list of hashes representing the history of previous versions of the DSU, suitable for signing and/or anchoring data (e.g., digital wallets).

4.2.2 Combined and complex DSUs

Obviously, a DSU may contain as data one or more KeySSIs that resolve to other DSU instances. In this light, we propose the design pattern of constSSIs, i.e., KeySSIs in a human-readable format that resolve to an immutable, read-only, "dummy" DSU containing a cryptographically more secure SSI that contains random numbers and therefore cannot be memorised intuitively anymore. The constSSI pattern circumvents a problem known as "Zooko's Triangle", according to which identifiers of any naming system cannot achieve all three attributes at the same time: meaningfulness to human, cryptographic security, and decentralisation (i.e., self-sovereignty) (Wilcox, 2001). Apart from pointing to each other by resolving contained KeySSIs, the DSU API also allows users to dynamically "mount" DSUs into one another, aggregating their contents but preserving their inherent access control. This enables a generic creation of nested containers that provide custom access levels across a possibly heterogeneous group of users.

For common use cases, OpenDSU additionally provides an already predefined class of "SecretDSUs" with a list of standard folders that can be accessed through different KeySSIs from the SecretSSI family: a "code" folder with the mounted DSU type; a "control" folder containing a whitelist of public keys that can modify the DSU, serving as an additional protection against attacks from trusted groups in possession of an AnchorID; a "public" folder containing the public key that provides basic access for holders of PublicSSI (or above) privileges; a "private" folder with confidential data readable only when employing a ReadSSI; a "secret" folder with private keys, e.g., for anchoring (i.e., AnchorSSIs); and an externally mounted "credentials" folder accessed using a CredentialSSI, with the signatures employed to validate versions of this DSU. As can be seen, CredentialSSIs do not stem from the SecretSSI family (Figure 3, lower left panel).

4.3 Near-chain data management

4.3.1 Brick storages

As introduced in the previous section, the contents of a DSU are assembled and decrypted dynamically from their persistent storage form, so-called "brick storages" (Alboaie et al., 2022c). Physically, a brick storage can store data on virtually any storage medium, e.g., locally, remotely, or in the cloud. Technically, a brick storage is a simple web service capable of storing and retrieving bricks for clients

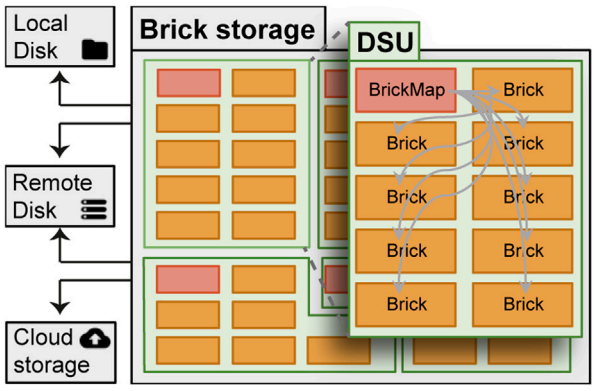

FIGURE 4
Brick Storage. Schema of a brick storage with bricks (orange boxes) and brick maps (red boxes) storing the DSU information off-chain either locally, remotely, or in a cloud.

(i.e., wallets) that know the identifying brick hash. Due to obvious security motivations, DSU data needs to be encrypted when exported to a brick storage. OpenDSU employs symmetric encryption, each brick is decrypted with its own particular KeySSI (Section 4.3).

Figure 4 outlines the brick representation of DSUs. To prevent undesired monitoring of changes, DSUs are disassembled into different bricks, in a process we call "bricking". Special bricks, that we refer to as "brick maps"—respectively "bar maps" in the legacy terminology of the earlier PrivateSky reference implementation of OpenDSU (Alboaie et al., 2020)—store the KeySSIs for the bucket of bricks they are referencing. As sketched in Section 4.1, a DSU during its life cycle is subject to changes, which may either alternate the content and thus also the hash identity of the DSU, or remove and respectively delete the DSU entirely. Despite these possibly high dynamics in modifications on a DSU, a brick map once created remains unchanged and referential by its hash in the persistent off-chain brick storage. Therefore, a new version of a modified DSU also requires a new brick map, and we refer to the list of all brick maps since the creation of a DSU as the corresponding DSU history.

Based on the bricking mechanism, we also developed BricksLedger, an OpenDSU component that is able to transform any database or blockchain into a ledger appropriate for OpenDSU anchoring (Alboaie et al., 2022f). On the one hand, BricksLedger provides a convenient drop-in for the use of standard databases or other existing data warehouses together with other ledgers in OpenDSU, achieving blockchain level audit for corresponding data. On the other hand, the possibility to anchor DSUs to virtually any data storage can effectively mitigate concerns about confidentiality of on-chain transactions and about performance (i.e., speed/throughput of the ledger).

4.3.2 Anchors

Calls to the DSU API (Section 4.2), e.g., writing files, keys and complementary functions, may modify the content (i.e., the "state") of a DSU, creating a new version of the DSU with a new ID that represents its public key. DSU modifications are made persistent by writing a new anchor to the blockchain, which comprises a hash code combining the new public key of the DSU as well as the DSU's history. Importantly, any write operation to a DSU will be observable by other execution environments only after the new version is successfully anchored, and all previous versions of a DSU are administered by a fully resolved history, signed by corresponding hashes. An anchor is composed by an identifier (i.e., AnchorID)—i.e., a special type of KeySSI identifying the DSU–and the history of brick maps from the respective DSU, represented by hash links also in the form of special KeySSIs. By this, OpenDSU anchors are self-validating, i.e., the DSU history of hash links in an anchor can only be truncated but not altered by manipulation. OpenDSU employs an anchoring technique to irrevocably bind DSUs to a blockchain, enabling the integrity and traceability of data, e.g., to verify near-chain data from a brick storage. Anchoring therefore allows for digitally signing data and code–in its initial version and all subsequent versions/updates (Alboaie et al., 2022h).

The sequence diagram in Figure 5 summarises the steps required for reading a file from a DSU, involving a number of complex operations that are orchestrated through a series of interconnected components. The process begins with a DSU read request (Figure 5 top) that is made to the OpenDSU SDK (Figure 2 top), which then transforms the request into an operation to read the corresponding AnchorID through an HTTPS call to the APIHub anchoring component (Figure 2 bottom). The anchoring component then sends the request to the respective blockchain by calling a method from a smart contract that has a list of all the hash links for all BrickMaps (i.e., all versions) of the respective DSU. Importantly, blockchain anchors may stem from different ledger domains, as resolved by the BDNS system (Section 4.1).

Subsequently, the smart contract returns the latest version of the BrickMap. A DSU can be compared to a big encrypted and compressed archive, however, its loading mechanism is "lazy" since only the by the operation requested bricks are retrieved. In contrast to less advanced techniques that retrieve the entire content of a DSU, lazily loading a DSU involves loading the BrickMap followed by reading merely one or more bricks. Importantly, only the BrickMap is decrypted during lazy loading, employing the SeedSSI that allows reading for access to the DSU (Section 4.2). The information from the bricks themselves is only decrypted in the execution environment, where it finally is displayed to the user (Figure 5 bottom).

After thorough research of the anchoring problem, we subdivide anchors into at least two types: implicit and explicit anchors. Implicit anchors are created intrinsically as a result of blockchain operations that are not explicitly stored in the world state (i.e., cache) of OpenDSU. To this end the immutable nature of blockchains serves as a notarisation mechanism for events or other specific information requiring "notarisation". However, OpenDSU avoids the usage of implicit anchoring by additionally providing explicit anchors implemented as smart contracts or in the form of some services outside of the ledger. By explicitly storing history and timestamps, explicit anchors are easy to reconstruct and to validate, as defined by anchoring smart contracts.

Explicit anchors are further divided into heavy and light anchors. Heavy anchors ensure that anchoring is controlled and

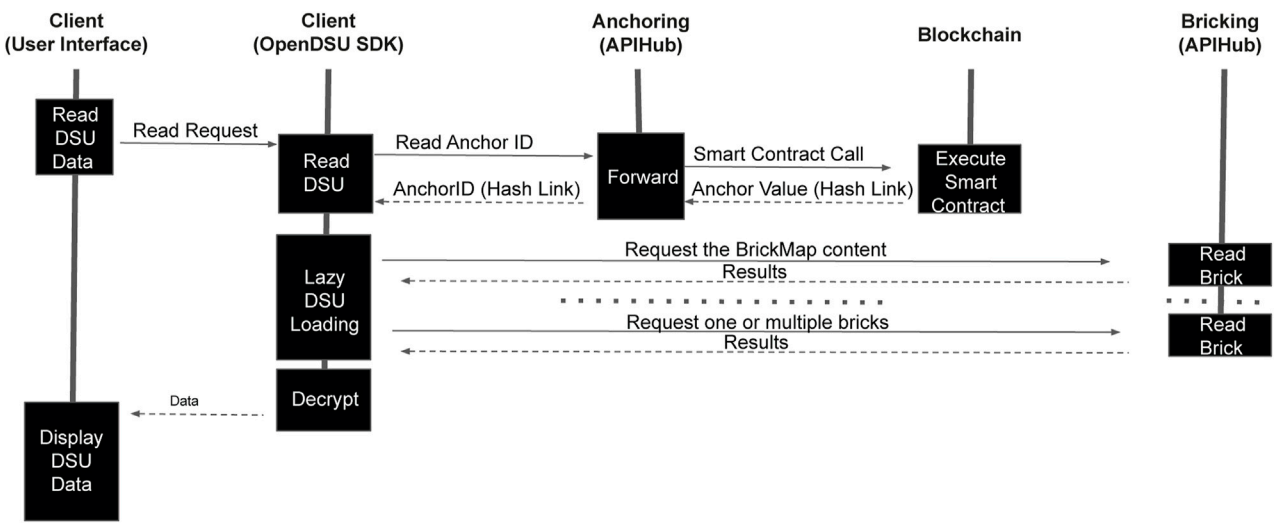

FIGURE 5
Example of resolving anchors in OpenDSU. The UML sequence diagram shows the process of lazily loading exclusively the required bricks of a DSU, after resolving the corresponding AnchorID. Of note, these AnchorIDs may originate from different ledger domains, uniformly resolved by the BDNS system (Section 4.1).

handled exclusively in smart contracts on-chain. In ledgers with support for smart contracts, they provide a costly though flexible solution considering the possibility of custom validations, the persistent hash history, and the timestamps. Light anchors, in contrast, require the on-chain storing of merely a list of hashes and a pair of anchor identifiers, obtained through zero access KeySSIs (Section 4.2). Anchored data as well as its history are in this case both reconstructed and validated from off-chain storages (Alboaie et al., 2022a). As an example for the philosophy behind our different anchoring strategies, we sketch here two ways of implementing Zero-Knowledge Proofs (ZKPs) in OpenDSU: on the one hand, ZKP protocols are traditionally implemented with implicit anchors, relying on the cryptographic properties of a blockchain. On the other hand, hash links could be implemented as explicit light anchors, accompanied by some custom cryptography to obtain ZKP values that can be used to prove computational integrity properties (i.e., zero access anchoring). Following the philosophy of not basing the privacy properties of a use case on a new and often not fully proven cryptography, the anchoring control in OpenDSU has been specifically conceived for light anchors, typically employing digital signatures. In use cases with an anticipated high number of anchors, light anchors also allow designing specialised high-throughput blockchains storing, e.g., billions or even trillions of light anchors (Section 5).

4.4 KeySSIs

Self-sovereign IDs across ledgers The DID (Section 2.1) community currently explores possibilities to overcome shortcomings in existing wallet technologies by developing novel methods for the communication between wallets that are capable of receiving credentials but also of transmitting and presenting VCs. A step ahead in this direction constitutes the DIDComm approach[15], which relies on DIDs that do not require blockchain anchoring, ensuring a very high level of privacy (Curren et al., 2022). However, W3C protocols like the DIDComm approach cannot be integrated well with our OpenDSU concepts. Firstly, they have been developed for home users and impose a rather unnecessary and complex overhead when implementing enterprise solutions (Section 5.1). Secondly, these anchor-less DID approaches cannot employ our BDNS functionality (Section 4.1). Although not mandatory in OpenDSU, KeySSIs are typically blockchain-anchored, enabling self-validation (Balagurusamy et al., 2019; Schillmann, 2020).

For OpenDSU, we therefore designed a blockchain agnostic through anchored SSI identifier concept through KeySSIs. As indicated by their name, KeySSIs are bifunctional: they can be used as keys for de-/encrypting (parts of) the information in DSUs or in bricks (Section 4.2), and at the same time they represent identifiers of their owners, who can be of physical (i.e., individuals), juridic (e.g., companies) or technical nature (e.g., resources/processes). Access privileges held by a certain KeySSI further can be delegated–entirely or in parts–to other KeySSIs derived from it, which subsequently are subtypes accompanying the original KeySSI. There are many possibilities to create KeySSI types/subtypes and we classify these into so-called “families”.

Like the DIDs proposed by the W3C standard proposal, KeySSIs can be controlled in a decentralised manner by possibly multiple domains. In this regard DIDs and KeySSIs can be considered two different methods for enabling SSIs (Section 2.1). The “type” field, as an equivalent to the DID “method” string, describes the specific (sub-) types of KeySSI families in OpenDSU, but provides an extended syntax to describe additional parameters as required in the OpenDSU

15 https://didcomm.org

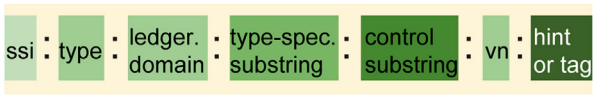

FIGURE 6
Syntax of KeySSI identifiers.

system. A very unique aspect of the KeySSI specification is field 3) that specifies the ledger/blockchain domain to which the SSI belongs, enabling an identity system that spawns multiple, possibly hierarchically cascaded ledgers respectively blockchains through BDNS (Section 4.1). Below a complete explanation of all KeySSI field descriptors summarised in Figure 6.

(1) Schema Identifier: "ssi" for keySSIs (cf. "Did" in the W3C standard for DIDs)
(2) Equivalent to the DID "method" string, KeySSIs are also categorised by a string describing their "type". SSI (sub-) types are compatible with each other and with the W3C DIDs, which allows a standard KeySSI resolver to implement any of these types.
(3) In addition to W3C DIDs, the SSI syntax provides a unique field to identify a blockchain or "ledger domain". Along the lines of the DNS system in the internet, OpenDSU introduces the "Blockchain Domain Naming System" (BDNS) to provide intelligible names for (sub-)networks, computers, end point services and users in–possibly hierarchical (Section 4)—blockchain networks (Alboaie et al., 2022b).
(4) The "type-specific substring" simply provides sufficient random bits for good security attributes.
(5) The "control substring" specifies the type 2). Specific algorithm used by anchoring services for validating and verifying requests for a new version of the anchored DSU.
(6) The "vn" string is optional (defaults to "v0") and informs the version number n of the KeySSI type 2) Not to be confused with the DSU versioning, a KeySSI type version specifies the cryptographic primitives and conventions, e.g., the hash functions and other methods to be used by the anchoring service (Alboaie et al., 2022f). New KeySSI type versions must be approved by the OpenDSU standardisation body, providing corresponding RFC documentation.
(7) The (optional) "hint or tag" string provides a generic way of extending KeySSIs for several purposes, again referring to the given (sub-)type 2). It typically "hints" at additional information for the KeySSI resolver, e.g., a favourite server proposed by the owner of the KeySSI. Alternatively, this string can be employed as a "tag" to mark KeySSIs for specific purposes; for instance DSUs that contain sensitive information will require additional data protection mechanisms in place.

4.4.1 KeySSI-based messaging through portable smart wallets

Section 1 outlines the progress of developments around web3, requiring ever more types of information in a digital wallet that cannot be provided by centralised wallet services. For instance, the currently popular Apple Pay[16] and Google Pay[17] wallets allow rather limited information to be "carried" inside them, e.g., credit cards, tickets for travelling or events, and other basic credentials, and cannot cope with DIDs or SSIs. The ability to flexibly represent user identities and at the same time to act as cryptographic keys opens up several possibilities of employing KeySSIs, and in order to overcome aforementioned shortcomings in existing digital wallets.

In OpenDSU, we therefore employ KeySSIs to implement a more simple yet effective protocol for communication between wallets. This OpenDSU messaging protocol employs the ECIES (Elliptic Curve Integrated Encryption Scheme) (Smart, 2001; Brown, 2009) to encrypt messages sent between DIDs based on KeySSIs in different scenarios of messaging (Brown, 2009). By this, KeySSIs can be exchanged to provide access to "messages" in the form of DSUs to which they resolve (Alboaie et al., 2022e). In addition, we employ AnchorIDs in notifications as names of (public) channels to provide "addresses" for DSUs that act as a mailbox (i.e., "message queues", MQs): everyone can write to a MQ, but only the respective owner can read the corresponding messages (Alboaie et al., 2022d). MQs can also be employed for automatic updates, when new versions of a DSU are anchored.

5 Results: the OpenDSU implementation in PharmaLedger

PharmaLedger is a multinational initiative, funded by the European Community and by IMI industry partners, to address several challenges of exchanging information in the pharmaceutical and in the health sector: sharing of patient records, communication between partners in the drug and medical equipment supply, etc. The PharmaLedger Project therefore includes public authorities, medical partners from the public sector, and commercial enterprises such as software companies and marketing authorisation holders (MAHs) of the pharmaceutical industry. The initial working package of PharmaLedger prioritised, from an initial list of >100 scenarios, seven use cases to be implemented within the pilot phase of the project, categorised into one of three so-called "domain reference applications" (DRAs).

- DRA1—"supply chains": e.g., the pharmaceutical supply chain (supply chain) use case, for tracking all the pharmaceutical items produced by each of the MAH partners, with an estimated number of up to billions of transactions per day;
- DRA2—"health data": e.g., the electronic product information (ePI) use case providing the MAHs' leaflets with information on their pharmaceutical products in an updatable form that granularly can be approved by the health authorities;
- DRA3—"clinical trials": e.g., the electronic patient consent (eConsent) use case that provides patients to dynamically provide respectively revoke by smart contracts their agreements on the use of their data in clinical studies;

16 https://www.apple.com/apple-pay/
17 https://pay.google.com

As sketched by these exemplary use cases across substantially different domains, PharmaLedger represents an environment where on the one hand data needs to be shared between different stakeholders, pointing towards a blockchain solution. On the other hand, MAHs' resources shared amongst PharmaLedger partners cannot always be stored in a common place, as intrinsic to all blockchain systems introduced in Section 3, but often have to reside at the premisses of the corresponding MAH. This requires a flexible nevertheless secure system of privileges, which allows MAHs the sovereignty to provide to collaborators access to their resources while obeying their particular business policies, but which also preserves at the same time the typical blockchain characteristics in the exchange of data.

Over the recent years, blockchain based initiatives have emerged all over the world to leverage digital sovereignty across different sectors, such as government, economics, energy and health (Alam et al., 2021). The OpenDSU SDK library (Alboaie et al., 2022i), originally implemented by the PrivateSky Project[18], has been improved by insights from different research as well as commercial projects, e.g., in the currently ongoing PharmaLedger Project[19]. The OpenDSU framework has been selected as a viable platform to develop the PharmaLedger solutions because its fundamental DSU concept allows to share data without necessarily replicating and/or relocating it to a centralised location (Section 4.2), but nevertheless maintains valuable blockchain attributes–such as the verifiability, the immutability and the auditability of shared data (Sec. 4.3). In the remainder of this section, we summarise the PharmaLedger contribution to the OpenDSU open-source project (Pritzker et al., 2021b), including use case validations, code adaptations, and proposals for further improvement.

5.1 DIDs in PharmaLedger

In order to provide PharmaLedger partners with the self sovereignty required for sharing their data with different collaborators across use cases, as outlined in the previous section, we investigated already in an early working package of the project the possibility of employing the DIDComm or other existing approaches to SSIs, which are based on the standards proposed by the W3C DIDs (Section 2.1). In a nutshell, our experiences led us to the conclusion that these standards have been developed for home users, who mostly aim at total anonymity and non-correlatability. However, the challenges of providing security and confidentiality of systems in an enterprise environment differ substantially from these issues of privacy protection.

More precisely, technologies along the lines of DIDComm optimise for enterprise environments that rely on VCs for creating trust. Consequently, issuers taking the role of a "root of trust" conceptually do not differ much from the approach of X.509 certificates. Although their more modern syntax and additional extensions may be beneficial from the confidentiality point of view, they also create an unnecessarily high degree of complexity, especially when considering ZKPs. Existing enterprise solutions for DIDs thus focus on circumstances that actually do not exist in a Digital Trust ecosystem such as PharmaLedger. We therefore estimated that adopting the current DIDComm approach to the real-world demands by the different PharmaLedger participants would infer an overhead that exceeds the concrete benefits.

In PharmaLedger, we had to segregate DID-based communication across different domains, as mapped by the BDNS topology of OpenDSU (Section 4.1). We therefore decided to develop during the project the keySSI system as described in Section 4.4 in order to work particularly with our BDNS strategy. This pragmatic approach allows us to adapt and integrate heterogeneous technical solutions for identifying patients (or citizens) across countries, without sticking to a pre-defined protocol that corrupts the flexibility of the OpenDSU-born BDNS. By this, unlike the approach predicated by the DID standardisation efforts, OpenDSU's solution to DIDs is not strictly focused on perfect standards, but provides a concept of solving the challenges of data sharing and segregation across sovereign domains.

Of note, OpenDSU wallets will still be able to adopt DIDComm or similar technologies along the same lines in the future, when standards have matured to overcome current limitations. The challenge of inter-country barriers caused by differences in the legal preconditions for health data policies is a paramount precondition for several PharmaLedger use cases, and has recently been leveraged by the General Directorate for Health and Food Safety of the European Council publishing a proposal on the regulation of the European Health Data Space (The European Health Data Space, 2022).

5.2 APIHub improvements for pharmaledger

We also assessed and identified components affecting the performance of OpenDSU in PharmaLedger. As introduced in Section 4.1, the different components of the APIHub together shape the networking layer (Layer 0), forming the main backend of an entire OpenDSU platform. When performing transactions, the APIHub further relies on a blockchain technology (Layer 1) that achieves the consensus. When benchmarking relatively early in the PharmaLedger Project an implementation of the ePI use case (Section 5) employing the Quorum blockchain[20], we identified three shortcomings of the original PrivateSky BricksLedger component (Section 4.3) that needed further improvement.

#1 *Latency*: a call on the Quorum blockchain takes a few seconds to be answered[21], s.t. a sequence of 2-3 calls, as required by most business use cases, causes delays of 5–10 s until the confirmation of a transaction and leads to a poor user experience.

18 https://profs.info.uaic.ro/~ads/PrivateSkyEn/

19 https://pharmaledger.eu

20 https://www.kaleido.io/blockchain-platform/quorum

21 https://consensys.net/quorum/

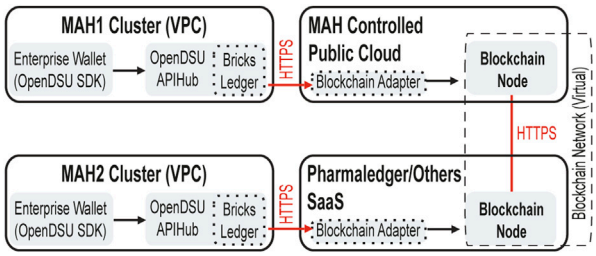

FIGURE 7
Deployment Clusters. Red arrows indicate the HTTPS communication paths between APIHub and blockchain nodes.

#2 *Throughput*: the scalability of the Quorum blockchain is limited to a few hundreds of transactions per second (Alboaie et al., 2022b).
#3 *Security*: custom PPP-based protocols for the communication of blockchain replica (i.e., DevP2P[22]) move MAHs to rely on weak security models, e.g., by employing a single cloud provider to host all blockchain nodes in a virtual network.

To alleviate observations #1 and #2, we directly decided to instantiate an APIHub separately for each of the PharmaLedger use cases. By this, transaction workloads are segregated, especially when accessing the blockchain. Next, in order to achieve the trustless aspects of blockchain technologies across the entire PharmaLedger OpenDSU platform, challenge #3 motivated us to maintain two deployment clusters: one for corporate internal transactions of MAHs, and another one spawning a blockchain network across consortial or public services. By this, each MAH needs to deploy and control a separate APIHub, requiring deployments different from the blockchain node. Like in the communication between blockchain nodes in OpenDSU, both "clusters" of APIHubs communicate via standardised and encrypted HTTPS, eliminating concerns about non-compliance with company security policies (Figure 7).

5.3 Optimistic execution of smart contracts

In due course of the PharmaLedger Project, we spent further efforts on researching solutions to mitigate challenge #1 (latency), while improving respectively not compromising challenges #2 and #3 (scalability and security, Section 5.2). According to our observations, PharmaLedger transactions are predominantly matching use cases of *Pattern #1* or *Pattern #2* (Section 2.3). As only very few partners are participating in such use case scenarios, the corresponding smart contracts may be executed in an "optimistic" way, i.e., in a way that is not immediately validated by the consensus algorithm. This led us to conceive an heuristic approach we call the "optimistic execution" of smart contracts (Anjana et al., 2021; 2019), which enables us to boost the performance of anchoring DSUs in PharmaLedger.

Figure 8 shows a layout of our concept of "optimistic execution", which attempts to alleviate the overall network overhead by reducing the workload for reaching a global consensus and instead resolving consensus straightforwardly in the "local" context of anchors. Our heuristics of executing such smart contracts "optimistically" relies on the fact that, in the absence of notarisation, it is possible to straightforwardly implement "self-consistent" anchoring commands: the OpenDSU anchoring command implicitly implements a "nonce-based mechanism", and each anchoring operation contains a signature binding it to the request with the last anchor variant, which *per se* is trusted to be correct by the signer.

Groups of anchors can more easily be trusted, especially when owned by a single owner of the corresponding DSU (use case *Pattern #1* and *Pattern #2*): to our observations in PharmaLedger use cases, most anchors are controlled by a single "producer" actor providing and updating (i.e., writing) data shared with "consumers". Therefore, "optimistically executed" smart contracts also are largely safe against common threats such as "replay" and/or "double spending" attacks. Moreover, since optimistic execution is by design intended to occur mostly in the nodes controlled by the owner of the respective anchors, also a possible issue by partitioning of the network is not a real issue in practice. In any case, implementing a validated execution, i.e., an execution that updates the validated world state as in the usual blockchain, will directly eliminate any potential issues.

Summarising all these considerations, the main advantage of the "optimistic execution" we propose is that the clients' requests can be responded very fast, because chances of the consensus invalidating the commands are zero in most of the PharmaLedger use cases. Additionally, the overall performance and scalability of the blockchain is improved by using this approach for anchoring.

5.4 Implementation of the ePI use case

In the PharmaLedger Project, the electronic Product Information (ePI) use case covers the access to leaflets by scanning the code on a product's package. This allows the manufacturer to dynamically attach relevant information to traditionally "static" leaflets, e.g., when calling back a specific batch of the production. In this light, the ePI use case aims to provide consumers with always the most up to date information on a product (read access), whereas MAHs require to periodically update and/or revoke parts of the information provided with their product (write access). In this section, we provide a brief overview of how we modelled this ePI use case in our OpenDSU platform.

Pharmaceutical products, as other commercial products, are uniquely identified by a Global Trade Item Number (GTIN), usually represented by a 1D or 2D barcode on the product package. When scanning this code with a mobile device, we employ a function (the so-called Key-Derivation-Function, KDF) to univocally generate from the GTIN an AnchorID of the OpenDSU platform. Following the design pattern of constSSIs sketched in Section 4.2, this AnchorID is human readable and resolves an invariant DSU that contains an sReadSSI reference to the DSU storing the actual

22 https://github.com/ethereum/devp2p

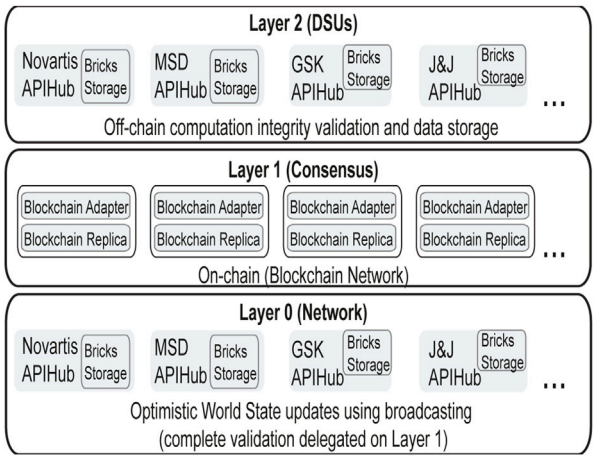

FIGURE 8
Logical view of the OpenDSU layers in PharmaLedger. Validation is delegated from the MAHs' APIHubs to the blockchain nodes (Layer 1).

information of the product. After assembly of latter DSU, the electronic product information (e.g., a PDF or XML document) is transferred to the mobile device that generated the request (Figure 9).

Of note, Figure 9 shows a simplified implementation of the ePI use case. In the PharmaLedger framework, we employ additional DSUs to cluster product DSUs by batches, in order to facilitate, for instance, the recall of a batch that affects multiple products. Furthermore, GTIN numbers do not provide enough entropy to derive cryptographically secure AnchorIDs. However, the use of GTIN-derived anchors is intentional as some applications in this use case foresee patients to be able to read leaflets identified by the GTIN of their associated product directly over the internet. To this end, we compensate the cryptographically weak key derivation from GTIN numbers by the immutable anchoring of the DSU storing the sReadSSI, preventing potential attacks that could aim at adding new versions or at modifying the content attached to the derived AnchorID.

6 Discussion

In the present paper, we introduce the concepts and components of OpenDSU, a generic platform that we developed over the past years and that enables the seamless integration of SSIs, DLTs and off-chain storage in enterprise environment use cases. The benefits of our platform are rooted in the design of conceptually sound components around central data sharing units (DSUs) that shuttle data and code to the endpoints where it is needed. By our universal BDNS interface, OpenDSU is agnostic to the respective technologies employed for spawning ledger networks, i.e., our platform supports the integration of arbitrary DLTs, including but not limited to blockchains.

Notwithstanding this flexibility of integrating even heterogeneous ledger technologies, OpenDSU provides a unified and seamlessly integrated concept for near-chain data, i.e., blockchain data exported to external premises that can be fully validated upon re-import to the ledger. We achieve this through bricking and anchoring, where bricks are atomary data blocks that can be stored in a privacy-preserving fashion on virtually any external medium, organised by blockchain-integrated anchors. Anchors coordinate the decomposition/reassembly of DSUs to/from bricks, maintaining previous versions of the DSU, which can be considered micro-ledgers with full access to their history of modifications.

Outside the execution environment where they are needed, all contents of a DSU are cryptographically secured through symmetric encryption and can be accessed only by employing an appropriate KeySSI. KeySSIs couple sovereign identity management with access control and can hierarchically be derived from one another, delegating some of the privileges from a higher to a lower access level. This combination further enables (automated) messaging through KeySSI identifiers, providing the completely self-sovereign and cryptographically secure exchange of information. Since KeySSIs are agnostic to the DLT(s) they are employed to, they pave the way for powerful cross-DLT applications in the future.

Recently, we successfully implemented our OpenDSU platform in the PharmaLedger Project, which brings together companies and public stakeholders from sectors as different as IT, healthcare and the pharmaceutical industry. As OpenDSU is conceived to employ generic components, without any lock-in model for the pre-defined components, we first conducted thorough research on the inclusion of existing solutions. Regarding the decentralised identity management, however, we pinpoint severe drawbacks by needless overhead and security concerns when employing current DID approaches to the heterogeneous business policies in PharmaLedger, because they have primarily been conceived for home users. Employing KeySSIs through the BDNS of OpenDSU in PharmaLedger allows the development[23] of substantially more flexible and scalable solutions to enable digital sovereignty across multiple ledger domains.

Conversely, our investigations on integrating existing blockchain technologies like Quorum in PharmaLedger also provided insights for improving OpenDSU components. To prevent security concerns that may arise from enterprise-internal data management, we distributed our backend across multiple APIHub instances, with specifically deployed APIHubs dedicated to the internal data management of each company. In this light, OpenDSU enables some benefits of distributed grid computing, where each MAH controls its own assets. Our multi-APIHub approach also permits setting up a separate APIHub for each use case in order to segregate the workload by high-throughput scenarios from the remaining operations. To additionally improve the latency and throughput of the system, we also developed an heuristic approach we call "optimistic execution" of smart contracts that resolves the consensus between a limited scope of participants more efficiently in a local context.

Beyond the scope of the present manuscript, we see future potential of OpenDSU's KeySSI-mediated messaging across

23 https://opensource.org/licenses/MIT

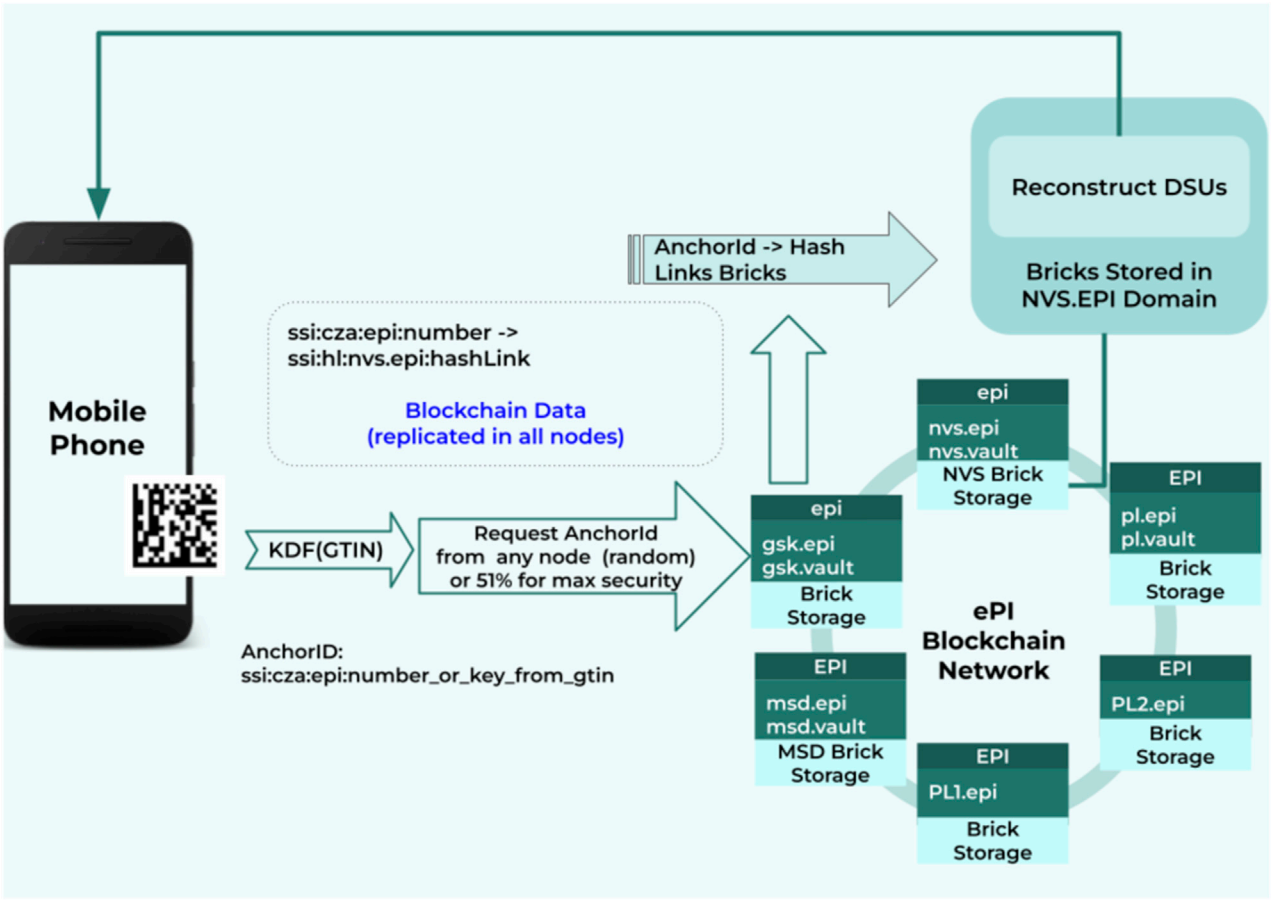

FIGURE 9
Flow diagram summarising how the electronic product information for a pharmaceutical product is obtained from the PharmaLedger OpenDSU platform: Scanning a code on the product package with a mobile device (*left*), generates an AnchorID by employing a key defining function (KDF) on the unique GTIN (*center*). This anchorID then is resolved to a DSU containing the corresponding leaflet information, stored in one of the brick storages (*bottom right*) maintained by a MAH (e.g., NVS = Novartis, GSK = GlaxoSmithKline, MSD = Merck Sharp & Dohme, etc.). Finally, the DSU with the product information is assembled and the electronic leaflet is transferred to the mobile device that originally scanned the code.

blockchain domains. As sketched in Section 4.4, message queues can be setup employing KeySSIs, and ongoing efforts aim at providing a generalised programming model, based on the concept of executable choreographies (Alboaie et al., 2013; 2014; Ursache, 2021). Unifying data sharing across use cases, such choreographies then could be formalised through recent efforts towards establishing general notations for document and data exchange patterns, by employing entity-relationship diagrams and/or UML patterns (Górski, 2022; Petrasch and Petrasch, 2022).

Moreover, the keySSI based communication between digital wallets in OpenDSU can provide a platform for developing modern protocols that obey the principles of privacy and security, and that may replace the email system as it existed since the dawn of the internet. As outlined in Section 4.2, DSUs can reference each other, effectively wrapping the corresponding SSIs employed for access. We thus propose to employ specifically encrypted encSSIs (encoded SSIs) as wrappers for KeySSIs exchanged over unencrypted channels. Such protocols could be designed from the beginning to be decentralised and difficult to centralise, overcoming current shortcomings by the monopolisation in the area of email services (Section 1).

7 Conclusion

In pursuit of building a Digital Trust Ecosystem for the pharmaceutical industry, the PharmaLedger project leveraged OpenDSU and witnessed notable success in meeting the requirements of the companies involved, because its concepts–particularly BDNS–offer clear advantages over complementary protocols based solely on DNS. Some critics have pointed out that OpenDSU's origin in European research projects instead of big corporations might constitute a disadvantage. However, the adoption of OpenDSU by PharmaLedger is a significant step in establishing a pedigree of nested albeit distinct policies across companies and countries.

It is challenging to compare OpenDSU with other existing approaches, as it offers new ways of organising the sharing of data between wallets along with new types of cryptographic keys.

In a nutshell, OpenDSU provides a generic framework for univocally modelling all the use cases required by the PharmaLedger consortium, whereas employing complementary platforms proposed to date essentially would require to build a custom solution for each use case. However, as the blockchain field is currently still continuously evolving, we conceived OpenDSU to be able to include other protocols, such as DIDComm, at a future point when standardisation will have matured. In this perspective, we focus with our future aims for the platform particularly on developing a light blockchain that is based on sophisticated methods of executing smart contracts optimistically and aims to solve exclusively the DSU anchoring.

The conclusion of all our ongoing research is that we are still only at the beginning of conceiving such future generation ledger systems. The fact that blockchain technologies currently are best candidates for handling secure transactions of NFTs and crypto-currencies in products envisioned to shape together the "Metaverse" highlights the need for an unprecedented interoperability of heterogeneous ledger networks. In this direction, we developed OpenDSU as a flexible platform to integrate largely arbitrary components for running ledger networks, blockchain anchoring, and the management of decentralised or self-sovereign identities. In the absence of lock-in mechanisms, OpenDSU is promoting a "Darwinian evolution" of the protocols rather than enforcing the standardisation of premature solutions. We conceived all OpenDSU components with the goal of enabling a broader acceptance of ledgers in the long run, by leveraging the full potential of technologies for digital sovereignty.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding authors. All OpenDSU code and documentation is freely available under the MIT open-source license.

Author contributions

N-CU: Conceptualisation, Methodology, Software, Validation, Visualisation, Writing—review and editing. MS: Funding acquisition, Investigation, Project administration, Visualisation, Writing—original draft, review and editing. SA: Conceptualisation, Funding acquisition, Investigation, Methodology, Project administration, Resources, Writing—review and editing. All authors contributed to the article and approved the submitted version.

Funding

This activity was financed by the European projects PrivateSky (Contract: 13/01.09.2016, SMIS 106611, ID P_40_371) and PharmaLedger (Grant agreement ID: 853992), all code and documentation is licensed under the MIT Open Source License.

Conflict of interest

Author N-CU and SA were employed by the company RomSoft SRL.

The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Alam, S., Shuaib, M., Khan, W. Z., Garg, S., Kaddoum, G., Hossain, M. S., et al. (2021). Blockchain-based initiatives: Current state and challenges. *Comput. Netw.* 198, 108395. doi:10.1016/j.comnet.2021.108395

Alblooshi, M., Salah, K., and Alhammadi, Y. (2018). Blockchain-based ownership management for medical iot (miot) devices. International Conference on Innovations in Information Technology (IIT). 18-19 November 2018, United Arab Emirates, IEEE, 151–156. doi:10.1109/INNOVATIONS.2018.8606032

Alboaie, L., Alboaie, S., and Barbu, T. (2014). Extending swarm communication to unify choreography and long-lived processes. *ISD2014 23rd Int. Conf. Inf. Syst. Dev. Transforming Organ. Soc. through Inf. Syst.* 382, 375. doi:10.1109/HPCC.and.EUC.2013.277

Alboaie, L., Alboaie, S., and Panu, A. (2013). Swarm communication - a messaging pattern proposal for dynamic scalability in cloud. IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing. November 13-15, 2013, USA, IEEE. doi:10.1109/HPCC.and.EUC.2013.277

Alboaie, S., Alboaie, L., Pritzker, Z., and Iftene, A. (2019). "Secret smart contracts in hierarchical blockchains," in *Information systems development: Information systems beyond 2020, ISD 2019 proceedings, toulon, France, august 28-30, 2019* A. Siarheyeva, C. Barry, M. Lang, H. Linger, and C. Schneider (Germany: ISEN Yncréa Méditerranée/Association for Information Systems), 1–12.

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022a). OpenDSU advanced: Anchoring (RFC-069), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc069 (Accessed 01 12, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022b). OpenDSU advanced: BDNS (RFC-067), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc067 (Accessed 01 12, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022c). OpenDSU advanced: Bricking (RFC-070), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc070 (Accessed 01 12, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022d). OpenDSU advanced: Message queues (RFC-073), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc073 (Accessed 06 24, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022e). OpenDSU advanced: Notifications (RFC-072), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc072 (Accessed 08 13, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022f). OpenDSU concepts: Anchoring (RFC-005), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc005 (Accessed 01 12, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022g). OpenDSU concepts: Brick storages (RFC-003), OpenDSU RFC documentation. Available at: Available at: https://opendsu.com/rfc003 (Accessed 01 12, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022h). OpenDSU concepts: DSU introduction (RFC-001), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc001 (Accessed 01 12, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022i). OpenDSU for developers: APIs overview (RFC-060), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc060 (Accessed 11 18, 2022).

Alboaie, S., Ursache, C., Lupu, T., and Tan, A. G. (2022j). OpenDSU for developers: DSU object (RFC-063), OpenDSU RFC documentation. Available at: https://opendsu.com/rfc063 (Accessed 01 12, 2022).

Alboaie, S., Ursache, N.-C., and Alboaie, L. (2020). Self-sovereign applications: Return control of data back to people. *Procedia Comput. Sci.* 176, 1531–1539. doi:10.1016/j.procs.2020.09.164

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., et al. (2018). "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*(New York, NY, USA: Association for Computing Machinery), '18, 1–15. doi:10.1145/3190508.3190538*EuroSys*

Anjana, P. S., Kumari, S., Peri, S., Rathor, S., and Somani, A. (2019). An efficient framework for optimistic concurrent execution of smart contracts. In 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). Feb. 15 2019, Italy, IEEE, 83–92. doi:10.1109/EMPDP.2019.8671637

Anjana, P. S., Kumari, S., Peri, S., Rathor, S., and Somani, A. (2021). Optsmart: A space efficient optimistic concurrent execution of smart contracts. arXiv:2102.04875v2. doi:10.48550/arXiv.2102.04875

Balagurusamy, V. S. K., Cabral, C., Coomaraswamy, S., Delamarche, E., Dillenberger, D. N., Dittmann, G., et al. (2019). Crypto anchors. *IBM J. Res. Dev.* 63, 1:1–4:12. 4. doi:10.1147/JRD.2019.2900651

Barclay, I., Radha, S. K., Preece, A. D., Taylor, I. J., and Nabrzyski, J. (2020). Certifying provenance of scientific datasets with self-sovereign identity and verifiable credentials. arXiv:2004.02796. doi:10.48550/arXiv.2004.02796

Blockchain Council (2022). Role of decentralized applications (dapps) in web 3.0 in 2022 and beyond. Available at: https://www.blockchain-council.org/web-3/role-of-dapps-in-web-3-0/(Accessed 03 13, 2023).

Brown, D. (2009). Sec 1: Elliptic Curve cryptography ver. 2.0. Available at: https://www.secg.org/sec1-v2.pdf (Accessed 11 25, 2022).

Corporation, I. B. M. (2018). Why new off-chain storage is required for blockchains. Available at: https://www.ibm.com/downloads/cas/RXOVXAPM (Accessed 10 19, 2022).

Curren, S., Looker, T., and Terbu, O. (2022). DIDComm Messaging v2.0. Available at: https://identity.foundation/didcomm-messaging/spec/v2.0/(Accessed 11 18, 2022).

Dixon, C. (2018). Why web3 matters. Available at: https://future.a16z.com/why-web3-matters/(Accessed 02 18, 2022).

Eberhardt, J., and Tai, S. (2017). "On or off the blockchain? Insights on off-chaining computation and data," in *Service-oriented and cloud computing*. Editors F. De Paoli, S. Schulte, and E. Broch Johnsen (Cham: Springer International Publishing), 3–15.

Edelman, G. (2021). The father of web3 wants you to trust less, wired. Available at: https://www.wired.com/story/web3-gavin-wood-interview/. Accessed: 2022-02-19

The European Health Data Space(2022). Directorate-general for health and food safety, proposal for a regulation. Available at: https://health.ec.europa.eu/publications/proposal-regulation-european-health-data-space_en. Accessed: 2022-07-15

Frankenfield, J. (2021). Hyperledger fabric. Available at: https://www.investopedia.com/terms/h/hyperledger-fabric.asp (Accessed 01 12, 2022).

Goldwasser, S., Micali, S., and Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM J. Comput.* 18, 186–208. doi:10.1137/0218012

Górski, T. (2022). Uml profile for messaging patterns in service-oriented architecture, microservices, and internet of things. *Appl. Sci.* 12, 12790. doi:10.3390/app122412790

Greenspan, G. (2015). Corda did method. Available at: https://www.multichain.com/download/MultiChain-White-Paper.pdf (Accessed 01 13, 2022).

Grigg, I. (2004). "The ricardian contract," in *Electronic contracting, 2004. Proceedings* (Germany: First IEEE International Workshop on IEEE, 25–31.

Hassan, S., and Filippi, P. D. (2021). Decentralized autonomous organization. *Internet Policy Rev.* 10 (2), 1. doi:10.14763/2021.2.1556

Ismail, L., and Materwala, H. (2019). Article A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions. *Symmetry* 11, 1198. doi:10.3390/sym11101198

Kirtani, P., Platt, M., and Solanki, N. (2020). Corda did method. Available at: https://nms.kcl.ac.uk/moritz.platt/assets/standards/2020-corda-did-method.pdf (Accessed 01 13, 2022).

Kubach, M., Schunck, C. H., Sellung, R., and Roßnagel, H. (2020). "Self-sovereign and decentralized identity as the future of identity management?," in *Open identity summit 2020*. Editors H. Roßnagel, C. H. Schunck, S. Mödersheim, and D. Hühnlein (Bonn: Gesellschaft für Informatik e.V.), 35–47. doi:10.18420/ois2020_03

Marlinspike, M. (2022). My first impressions of web3. Available at: https://moxie.org/2022/01/07/web3-first-impressions.html (Accessed 02 18, 2022).

Nakamoto, S. (2018). Bitcoin: A peer-to-peer electronic cash system. Available at: https://bitcoin.org/bitcoin.pdf (Accessed 01 13, 2022).

Petrasch, R. J., and Petrasch, R. R. (2022). Data integration and interoperability: Towards a model-driven and pattern-oriented approach. *Modelling* 3, 105–126. doi:10.3390/modelling3010008

Pritzker, Z., Alboaie, S., Cuomo, M., and Patsonakis, C. (2021a). D3.1 pharmaledger framework architecture. Available at: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5dc522cc4&appId=PPGMS.

Pritzker, Z., Alboaie, S., Mastahac, B., and Balan, A. (2021b). D3.10 first reference implementation of pharmaledger platform. Available at: https://www.slideshare.net/PharmaLedger/first-reference-implementation-of-pharmaledger-governance-ui. URI Ares 7817797 (Accessed 12 17, 2021).

Pritzker, Z., Liappas, N., Arman, M., Raihan, M., Patsonakis, L. M., Christos, H., et al. (2021c). D3.3 blockchain platform research. Available at: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5dc523623&appId=PPGMS (Accessed 01 13, 2022).

Rahman, M. S., Chamikara, M., Khalil, I., and Bouras, A. (2022). Blockchain-of-blockchains: An interoperable blockchain platform for ensuring iot data integrity in smart city. *J. Industrial Inf. Integration* 30, 100408. doi:10.1016/j.jii.2022.100408

Sabt, M., Achemlal, M., and Bouabdallah, A. (2015). Trusted execution environment: What it is, and what it is not. *IEEE Trustcom/BigDataSE/ISPA* 1, 57–64. doi:10.1109/Trustcom.2015.357

Schär, F. (2021). Decentralized finance: On blockchain- and smart contract-based financial markets. *Fed. Reserve Bank St* 2021, 153–174. doi:10.20955/r.103.153-74

Schillmann, C. F. (2020). Crypto anchors in digital supply chain management - what value does it offer? *Msc. International management*. Germany: Nottingham University Business School. doi:10.13140/RG.2.2.12868.24960

Smart, N. P. (2001). "The exact security of ecies in the generic group model," in *Cryptography and coding*. Editor B. Honary (Berlin, Heidelberg: Springer Berlin Heidelberg), 73–84.

Sporny, M., Longley, D., and Chadwick, D. (2021a). Verifiable credentials data Model 1.0, w3c recommendation. Available at: https://www.w3.org/TR/vc-data-model/. Accessed: 2022-01-19

Sporny, M., Longley, D., Sabadello, M., Reed, D., Steele, O., and Allen, C. (2021b). Decentralized identifiers (dids) v1.0, world wide web consortium (w3c). Available at: https://www.w3.org/TR/did-core/(Accessed 01 22, 2022).

Tanner, J., and Khan, R. (2021). Technology review of blockchain data privacy solutions. arXiv:2105.01316. doi:10.48550/arXiv.2105.01316

Thakkar, P., Nathan, S., and Viswanathan, B. (2018). Performance benchmarking and optimizing hyperledger fabric blockchain platform. IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems 25-28 Sept. 2018, Germany, (MASCOTS 264–276. doi:10.1109/MASCOTS.2018.00034

The Federal Reserve Bank of Boston (2019). Beyond theory: Getting practical with blockchain. Available at: https://www.bostonfed.org/-/media/Documents/one-time-pubs/2019/blockchain-white-paper.pdf (Accessed 11 11, 2021).

The Linux Foundation (2020). Hyperledger fabric. Available at: https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf (Accessed 11, 2021).

Ursache, N.-C. (2021).Swarm communication using self sovereign identities, 2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet), 4-6 Nov. 2021, USA, IEEE. 1–6. doi:10.1109/RoEduNet54112.2021.9638293

Viriyasitavat, W., Bi, Z., and Hoonsopon, D. (2022). Blockchain technologies for interoperation of business processes in smart supply chains. *J. Industrial Inf. Integration* 26, 100326. doi:10.1016/j.jii.2022.100326

Wilcox, B. Z. (2001). Zooko's Triangle, names: Distributed, secure, human-readable: Choose two. Available at: https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html (Accessed 10 16, 2019).