Computers & Security 146 (2024) 104050

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

A privacy-aware authentication and usage-controlled access protocol for IIoT decentralized data marketplace

Akanksha Dixit [a,*], Bruno Bogaz Zarpelao [b], Max Smith-Creasey [c], Muttukrishnan Rajarajan [a]

[a] School of Science and Technology, City, University of London, UK
[b] Computer Science Department, State University of Londrina (UEL), Brazil
[c] IEEE UK and Ireland Cyber Security Group, UK

ARTICLE INFO

Keywords:
Data marketplace
Authentication
Usage control
Decentralized Identifiers (DID)
Verifiable Credentials (VC)
Privacy
User-revocation

ABSTRACT

Data is ubiquitous, powerful and valuable today. With vast instalments of Industrial Internet-of-Things (IIoT) infrastructure, data is in abundance albeit sitting in organizational silos. Data Marketplaces have emerged to allow monetization of data by trading it with interested buyers. While centralized marketplaces are common, they are controlled by few and are non-transparent. Decentralized data marketplaces allow the democratization of rates, trading terms and fine control to participants. However, in such a marketplace, ensuring privacy and security is crucial. Existing data exchange schemes depend on a trusted third party for key management during authentication and rely on a 'one-time-off' approach to authorization. This paper proposes a user-empowered, privacy-aware, authentication and usage-controlled access protocol for IIoT data marketplace. The proposed protocol leverages the concept of Self-Sovereign Identity (SSI) and is based on the standards of Decentralized Identifier (DID) and Verifiable Credential (VC). DIDs empower buyers and give them complete control over their identities. The buyers authenticate and prove claims to access data securely using VC. The proposed protocol also implements a dynamic user-revocation policy. Usage-controlled based access provides secure ongoing authorization during data exchange. A detailed performance and security analysis is provided to show its feasibility.

1. Introduction

Sensing-as-a-service is a concept that allows data owners to sell and/or exchange sensor data with consumers interested in large open markets. Data owners with the capability to generate vast amounts of sensor data engage in such a data exchange primarily for monetization purposes. A data marketplace is an open platform that allows buyers to look up, browse and purchase the most suitable data according to their needs from a vast array of heterogeneous data hosted on it by various sellers in domains like health, automotive, manufacturing, agriculture (Jeedigunta et al., 2023), etc. Unlike centralized data marketplaces (Oracle, 2023), which are governed and managed by a handful of central authorities, a decentralized data marketplace leveraging distributed ledger technology (DLT) like blockchains is more transparent, enabling sellers to set their rates and terms of data exchange. Aiming to explore these benefits, multiple studies such as (Dixit et al., 2023) and Qi et al. (2022) have attempted to design such marketplaces for data sharing. A decentralized data marketplace based on DLT is essentially a trading portal implemented as a DApp with the help of blockchain smart contracts and a decentralized data streaming network to host IoT data in a reliable and fault-tolerant manner. DApp is a decentralized software application that interacts with the blockchain back-end using smart contracts. The smart contracts perform functions such as listing, onboarding actors, enabling data upload and other business logic related to the trading of data. The data streaming network is hosted by a cluster of decentralized peer-to-peer (P2P) nodes i.e. *storage operator* nodes that host IoT data on behalf of *sellers* as they may not have the infrastructure or will to do so. *Storage operator* nodes host the data in the form of data streams and are incentivized for their service.

While decentralized IoT data marketplaces have enabled sellers and buyers to interact and engage in data trading in a fair and transparent manner, security and privacy concerns exist. To highlight the first concern, the conventional state-of-the-art marketplaces deploy a centralized identity infrastructure which gives them unrestricted view to user's activities on their platform. While centralized identity systems are easy to maintain and deploy, they have serious security challenges like a single point of failure, lack of user control, data breaches, linkability and eventually privacy threats. Additionally, a malicious

* Corresponding author.
E-mail addresses: akanksha.dixit@city.ac.uk (A. Dixit), r.muttukrishnan@city.ac.uk (M. Rajarajan).

https://doi.org/10.1016/j.cose.2024.104050
Received 1 September 2023; Received in revised form 12 June 2024; Accepted 6 August 2024
Available online 8 August 2024

storage operator can easily trace the activity of a *buyer* and gather personal details while transacting on the marketplace. Tracking the activities of a *buyer* allows the attacker to obtain their digital footprints and understand their purchase patterns. Marketing and advertising companies can target the *buyer* to send unsolicited advertisements or even influence their choice when browsing the marketplace. This is a familiar trend in centralized data marketplace where the marketplace operator has full knowledge of the activities of every seller and buyer on their platform.

The second security concern lies in the fact that the existing data marketplace authentication protocols do not consider the importance of *buyer* empowerment; the user is not at the centre of such designs. Traditional designs deploy a trusted party in a centralized system to manage the identities and associated keys of *sellers* and *buyers* which leads to the risk of information leakage. In such systems, the exposure of hundreds of user data is only a hack away.

In a data marketplace, the *buyer*, i.e., the user, needs to be authenticated and authorized in a secure manner by the *storage operator* to establish their claims and rights to access data. The third security concern in a data marketplace is that a malicious *buyer* can eavesdrop on the communication and use the obtained credential to get authorization that works in a 'one-time-off'. 'One-time-off' format is witnessed in traditional authorization schemes such as Role-Based Access Control (RBAC) or Attribute-Based Access Control (ABAC). Such access control models are tightly coupled to the user's identity to perform authorization only once before the access is allowed and do not continuously monitor the usage of its hosted resource or evaluate change in the user' s attributes during resource usage. While traditional access schemes work well for static data like video content, files, images, etc., IoT generates real-time data that draws its value from its *freshness*. Thus, in a real-time data access scheme for a data marketplace, continuous usage-based monitoring is required. In addition, a dishonest *buyer* who failed to complete previous payments should be barred from accessing any more data streams. Therefore, ensuring the security and privacy of all the parties in the marketplace is essential.

In the proposed work, we focus on the above-discussed issues of an IoT data marketplace. The discussed challenges in a data marketplace need a new approach to how identities are established and claims are presented and verified between the user and verifier. We build upon the concept of decentralized identity in this work to solve these challenges. While the overall security of the system is crucial, maintaining the privacy of the users is paramount too. DIDs and derived pseudo identifiers in the proposed framework ensure the credibility of the user without disclosing any PII (personally identifiable information) related to them. When these DIDs are bound with the issued VCs from an authorized issuer, they can convey the user' s claims in a privacy-preserving way to the verifier. Hence, data minimization is ensured in all the communications between the issuer, holder and verifier. ZKP-enabled VC will allow the holder to share only the proof of their claims with the verifier. Hence, the privacy of the user is maintained and the digital footprint is reduced. Dynamic revocation in a marketplace is crucial so that malicious actors cannot continue to operate and cause any harm to the rest of the users/services. Finally, continuous monitoring of the data streams being accessed is crucial for two main reasons, (i) to ensure fair compensation to the seller and storage operator for the data and (ii) to lock out malicious actors from further accessing the data streams.

This paper proposes a user-empowered, privacy-aware, authentication and usage-controlled access protocol for a decentralized IoT data marketplace. The proposed protocol leverages the concept of SSI to enable user empowerment. To tackle the challenge of privacy preservation, remove reliance on a centralized key-managing system, and establish trust among involved parties, in this paper we leverage the concept of DID (Sporny et al., 2023) and VC (Sporny et al., 2022), two main standards of SSI. Though DID and VC have emerged as promising technologies to enable decentralization and user empowerment, they have not addressed user revocation. Dishonest *buyers* should be removed from the system, thus the proposed protocol implements a dynamic *buyer* revocation. We also provide efficient Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) proof, which is a privacy-preserving way to present and verify VC. After secure authentication, the *buyer's* claims to access the data stream is evaluated against its associated policies and on a successful evaluation, access is allotted. Thereafter, the usage of resources is monitored on an ongoing basis and policies are continuously evaluated until the data consumption is completed. In the proposed paper, after successfully establishing the legitimacy of the *buyer*, the *storage operator* runs a usage-controlled access protocol until the completion of data consumption by the *buyer*. The proposed work is an extension of our earlier work (Dixit et al., 2023) that presented a fair, secure and trusted IIoT data marketplace. It is worth noting that the work in Dixit et al. (2023) focuses on aspects of IoT data marketplace like defining architecture and processes that assure fair settlement of compensation between parties, enabling unbiased content listing using calculated trust metric, converting business logic into an automated set of transaction rules for trading and fault tolerant and assured data delivery. It does not focus on other aspects of a data marketplace like security and privacy of users in such a marketplace, secure authentication and authorization-related flows that provide granular monitoring of resources being accessed on the marketplace. The latter is the focus of the work presented in this paper. The key contributions of the proposed work are the following:

- **A novel buyer-empowerment based privacy-aware authentication protocol based on DID and VC:** the proposed protocol enables buyer' s autonomy and eliminates reliance on a centralized identity infrastructure. Buyers present Zero-Knowledge proof (ZKP) of their VCs to the *storage operator* in order to access data.
- **A usage-controlled access protocol to monitor usage and buyer attributes on an ongoing evaluation basis:** the proposed protocol evaluates access control decisions based on continuous monitoring of IoT data resource usage to provide efficient and secure control. *Storage operator* nodes run this protocol on their end to control access to hosted IoT data resources to *buyers*.
- **A dynamic buyer revocation policy**: buyers on the platform who are dishonest, have non-payment dues or violated terms of data exchange are put on the revocation list. A *buyer* must prove to the *storage operator* that they are not on the revocation list to request data stream access. This is achieved in a privacy-aware manner.
- **Proof-of-concept (PoC) implementation and Evaluation**: a PoC is implemented with wallets using Hyperledger Indy Blockchain and a decentralized real-time usage-controlled access policy is implemented using a Linux utility: Cronjob. The paper includes a discussion of performance and security analysis.

The rest of this paper is organized as follows: Section 2 describes the related work. Section 3 discusses the fundamental concepts used in the proposed protocol design. Section 4 presents the proposed protocol in detail including step-wise discussion of each phase of the protocol. Section 5 describes the experimental settings and discusses the implementation results for each stage of protocol execution. Section 6 presents the security analysis of the protocol. Section 7 presents the conclusion of the paper.

2. Related work

Blockchain-enabled data sharing: data exchange facilitated by blockchain has garnered significant attention due to inherent benefits like non-repudiation, and providing a trusted sharing service, where information is reliable and can be traced (Reyna et al., 2018). In Lu et al. (2022), authors present a blockchain-based cloud storage protocol for sensors in IIoT using a group signature scheme. Smart contracts and proxy re-encryption are used to realize data sharing in an industrial facility. A blockchain-enabled dynamic and traceable data-sharing

scheme for the smart factory is proposed in Ma et al. (2023). Blockchain performs the authentication and stores the ciphertext index and public keys to avoid tampering. The tracking algorithm tracks malicious users and adds them to a revocation list. The authors in Qi et al. (2021) propose a private data-sharing framework using blockchain for applications such as tracking products for counterfeit detection, product recall and compliance with regulations. A blockchain-based hierarchical data-sharing framework to provide fine-grained access control and retrieval over encrypted personal health records between multiple stakeholders is presented in Zhang et al. (2022b). The authors in Zhang et al. (2022a) propose a monitor-based usage control model to enforce data usage policies on the user side using blockchain and SGX (Software Guard Extensions). The authors in Koutsos et al. (2020) and Koch et al. (2022) have presented privacy-aware data marketplaces that focus on the security of data and privacy of data sellers. While seller privacy is an important aspect of any data marketplace design, it is increasingly important for healthcare-related data as presented in the cited work. The proposed work focuses on industrial IoT settings, where large organizations typically act as data sellers. In such cases, privacy requirements for sellers might be relaxed. Nevertheless, the same principles devised for buyers in the present work can be extended to sellers whenever needed.

Though the above-discussed schemes ensure the secure exchange of IoT data and while some of them also focus on the privacy of the sensors, most of them use a centralized model for data exchange and do not consider a multi-stakeholder environment. Some of them store the data on blockchain and most of them use complex key-distribution mechanisms. In some of these schemes, the system master secret key was generated in a centralized manner by a trusted party. If a trusted party is compromised by an adversary, there is the risk of leakage of the system master key. Further, most of them have considered a 'one-time-off' access control technique that does not protect against insider attacks during resource usage.

IoT Data Trading: on the other hand, literature on IoT data *trading* or Data-as-a-Service model for monetization has also seen progress (Oh et al., 2020; Tian et al., 2019). Such systems mostly involve a P2P data sharing model where the data owner can decide the rates, terms and policies for data trading (Banerjee et al., 2020), Dixit et al. (2023). Figueredo et al. (2022) discuss one such practical IoT data marketplace framework 'oneTRANSPORT' implemented between various public bodies in the UK. A blockchain-based solution for resource-constrained IoT streaming devices that allows data chunks to be transferred in a decentralized, traceable and secure manner has been proposed in Hasan et al. (2022). Qi et al. (2022) present a 'Databox-based' delivery service via blockchain to provide data consumers with secure and controlled access to the requested data source of interest. Trading of IoT data in other domains like health (Agbo et al., 2019) and energy data (Cui et al., 2022) has also witnessed interest. Though the above-mentioned works try to address different issues related to an IoT data trading marketplace, they lack details on users' privacy, use empowerment and most importantly monitoring IoT data usage when it is traded on such decentralized platforms. User authentication, identity verification and usage-control layer are missing from the above-mentioned proposals. Proving the access rights and user's claims in a privacy-preserving manner remains unaddressed in the state-of-the-art decentralized data marketplaces.

Use of DID in IoT: the concept of DID and VC has recently gained popularity as a viable privacy-preserving method to enable identity empowerment in IoT (Kyriakidou et al., 2023), Sarower and Hassan (2023). The use of DID for IoT applications was discussed in Kortesniemi et al. (2019). The work indicates the feasibility of DID even on devices with limited resources. IoT-as-a-Service (IoTaaS) concept to offer IoT device services using DID and VC on a demand basis has been proposed by Diego et al. (2021). A solution based on DID and VC that enables a distributed V2X (Vehicle to everything) access authorization mechanism to allow a vehicle owner to prove their identity without compromising privacy was proposed in Lim et al. (2023). While DID enables authentication and user identification, VC can verify a set of claims that can be verified against access policy to allow secure and granular authorization (Saidi et al., 2022). Oku et al. (2022) present a privacy-sensitive information protection and management scheme to allow automated distribution of information. Though the above-discussed schemes employed DID and VC to allow users to control their identity, they have not considered key issues like dynamic user revocation. The schemes in Saidi et al. (2022) and Oku et al. (2022) have not considered the critical issue of continuous resource usage monitoring in an access control scheme.

Fig. 1. A DID Document using JSON representation.

The above-discussed works clearly outline the importance of IoT data trading and the use of blockchain technology for such data exchange/trading applications. Blockchain offers several advantages in a multi-stakeholder environment where parties inherently do not trust each other and wish to achieve transparency, and non-repudiation and avoid siloed views of transactions. A few of these data trading applications also propose the introduction of DID and VC as identity pillars to remove the centralization of identity management. To fulfil the above-identified research gaps, we propose a privacy-aware, authentication and usage-controlled access protocol for IIoT data marketplaces. In the proposed work we not only leverage the use of DIDs and VCs for providing mutual authentication between the parties coming together to trade on such a platform, we do so in a privacy-enhancing way so that tracking of engaging parties is not feasible for the marketplace service provider. This inhibits behaviour such as exhibiting biased search results, targeted ads or correlation of a user's activities on the marketplace. The traceability is sufficiently difficult to achieve with the help of DIDs. Our solution also provides anonymity for trading parties in case they wish to do so using ZKP-enabled VC. Our protocol makes dynamic revocation of dishonest parties from the marketplace so that they can no longer engage in trading activities. Once mutual authentication between trading parties is achieved, our protocol continuously monitors user and resource attributes to achieve a more secure transaction using usage control. A feature-wise comparison of the most closely related works discussed above is presented in Table 1.

3. Preliminaries

This section discusses the cryptographic and mathematical concepts used in the proposed protocol.

Decentralized Identifiers: DID is a type of globally unique identifier that is designed to provide verifiable, decentralized digital identities to a user, organization or thing (Sporny et al., 2023). No central authority is involved in issuing or maintaining a DID which gives the holder complete control and ownership over their IDs. A holder can create different DIDs for interaction with different parties to ensure unlinkability by correlation of their online activities. Any decentralized network, such as blockchain, can be used to resolve a unique key to a unique value.

DIDs are underpinned by asymmetric cryptography that uses a public–private key pair. The private keys are used to sign messages by the DID owner while the public key is available to verify signed

Table 1
Comparison of blockchain marketplace solutions based on design features [P = *Proposed work*].

Design Features	Hasan et al. (2022)	Oh et al. (2020)	Oku et al. (2022)	Qi et al. (2022)	Qi et al. (2021)	Tian et al. (2019)	Diego et al. (2021)	P
Privacy of User	–	+	+	–	+	+	+	+
User empowerment	–	–	+	–	–	–	+	+
Mutual Auth	–	–	–	–	–	–	–	+
Decentralized ID	–	–	+	–	–	–	+	+
Dynamic Revocation	–	–	–	–	–	–	–	+
Non-Repudiation	+	–	+	+	+	+	+	+
Unlinkability	–	–	+	–	–	–	+	+
Multi-Stakeholders	+	+	–	+	+	+	+	+
Access-Policy	–	–	+	+	+	–	–	+
Usage-Control	–	–	–	–	–	–	–	+

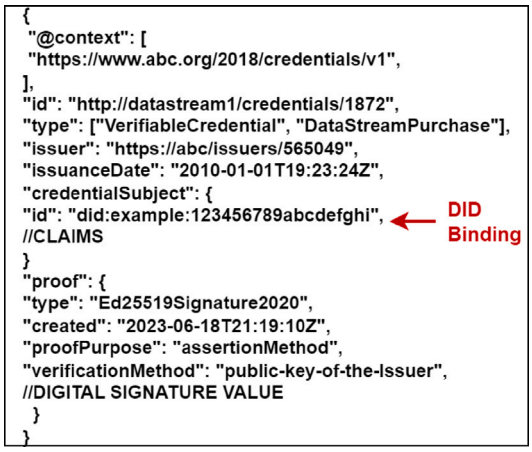

Fig. 2. A Verifiable Credential signed with EdDSA Signature.

messages. A DID resolves into a DID Document using a DID Resolver and this process of mapping a DID to its DID Document is called *resolution*. A DID supports four common CRUD operations: 'Create', 'Read', 'Update' and 'Deactivate'. The 'Read' operation is the most frequently used operation and is used by the DID Resolver to resolve a DID to its DID Document. DID has the following syntax: *"did: < DIDmethod >: < method − specific identifier >"*. The DID Method is a reference to the underlying distributed ledger network. A unique identity in that distinct network is defined by the method-specific identifier and helps in resolving the DID to a DID document on the ledger. An example of a DID document is shown in Fig. 1.

Verifiable Credential: In a physical world, a physical credential such as a driver's license card or a passport contains information related to the identifying subject. It claims certain rights that the holder can ascertain. For example, a driver's license proves to the verifier the holder's ability to drive a certain vehicle, their name, age, etc. Similarly in the digital realm, a VC can represent all of the same information that a physical credential represents. The addition of technologies, such as digital signatures, makes VCs more tamper-evident and cryptographically verifiable thus more trustworthy than their physical counterparts. The VC ecosystem mainly consists of three major actors; *Issuer*, *Holder* and *Verifier*. The **issuer** attests and issues credentials (a set of issued attributes) about the VC **holder** and includes proof in this VC such as a digital signature generated using its private keys. When a VC is presented to a **verifier**, it can verify the attributes/claims about the **holder** using the public key of the issuer (Sporny et al., 2022). Since the digital signature of the issuer is used as proof embedded in the VC, they are secure and digitally verifiable. A trusted *issuer* issues a signed VC to the *holder* and binds it to their DID after verifying their identity as shown in Fig. 2. To access a service/resource held by a *verifier*, the user reveals their DID and the issued VC to the *verifier*. The *verifier* verifies the claims in the VC presented by the holder using the *issuer's* public key. In this setting, verification does not depend on the direct interaction between the issuer and verifier, thus decoupling both entities.

Table 2
ZKP algorithms.

Algorithm	Description
$(crs,\ t) \leftarrow\ Setup(R)$	For relation R, crs and t are generated.
$\pi \leftarrow Prove(R, crs, st, w)$	This algorithm takes crs and $(st, w) \in R$ as input and return π.
$0/1 \leftarrow Verify(R, crs, st, \pi)$	crs, st and π are the inputs for this algorithm and 0(reject) or 1(accept), is the output.
$\pi \leftarrow Sim(R, t, st)$	The simulator takes t and st as inputs and returns π.

Zero-Knowledge Proof (ZKP): ZKP is a cryptographic way of presenting knowledge that a *Prover* wants to prove to the *Verifier* without revealing the knowledge itself. The Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) is a ZKP construct, which a Prover can use to prove knowledge of information to the Verifier without revealing the information and without any interaction between them. The zk-SNARK system presented in Groth (2016) is the most efficient and widely accepted, therefore, we use this in the proposed protocol. A common reference string is shared between the Prover and Verifier to achieve ZKP (Groth, 2016). In the algorithms presented in Table 2, a relation generator $\mathbb{R}$ returns a binary relation R for a security parameter λ. For pairs $(st,\ w) \in R$, st is the statement, w is the witness, crs is the common reference string and t indicates the simulation trapdoor. An efficient prover publicly verifiable non-interactive argument for $\mathbb{R}$ is a quadruple of probabilistic polynomial algorithms (Setup, Prove, Verify, Sim) as shown in Table 2 such that:

- $(crs, t) \leftarrow Setup(R)$: the setup produces a common reference string crs and a simulation trapdoor t for the relation R.
- $\pi \leftarrow Prove(R, crs, st, w)$: the prover algorithm takes as input a common reference string crs and $(st, w) \in R$ and returns an argument π.
- $0/1 \leftarrow Verify(R, crs, st, \pi)$: the verification algorithm takes as input a common reference string crs, a statement st and an argument π and returns 0 (reject) or 1 (accept).
- $\pi \leftarrow Sim(R, t, st)$: The simulator takes as input a simulation trapdoor and statement st and returns an argument π.

There are three properties of ZKP (Groth, 2016): completeness, zero-knowledge and soundness.

Completeness: An honest prover can convince a true statement to an honest verifier:

$$Pr[(crs, t) \leftarrow Setup(R); \pi \leftarrow Prove(R, crs, st, w)\ :$$

$$Verify(R, crs, st.\pi) = 1] = 1.$$

Zero-Knowledge: The proof does not reveal anything other than the truthfulness of the statement. For all $\lambda \in N$, $(R, z) \leftarrow R(1^{\lambda}), (st, w) \in R$

and adversary A, we can write:

$$Pr[(crs, t) \leftarrow Setup(R); \pi \leftarrow Prove(R, crs, st, w) :$$

$$A(R, z, crs, t\pi) = 1]$$

$$= Pr[(crs, t) \leftarrow Setup(R); \pi \leftarrow Sim(R, t, st)]$$

$$: A(R, z, crs, t, \pi) = 1.$$

Soundness: A prover cannot prove a false statement to the verifier:

$$Pr[(R, z) \leftarrow R(1^{\lambda}); (crs, t) \leftarrow Setup(R);$$

$$(st, \pi) \leftarrow A(R, z, crs) :$$

$$st \notin L_R \wedge Verify(R, crs, st, \pi) = 1] \approx 0.$$

Dynamic Cryptography Accumulator: an accumulator is a short binding commitment to a set of elements and allows for a short (non)membership proof for any element in the set or not in the set. These proofs, also called witnesses (witness to element being accumulated in the accumulator), can be verified against the commitment (Camenisch and Lysyanskaya, 2002). A dynamic accumulator that allows addition/removal of values dynamically is proposed in Li et al. (2007). The proposed protocol implements a revocation policy using a dynamic universal accumulator to remove dishonest or malicious *buyers* from the marketplace platform. A *buyer* needs to provide a non-membership witness that they are not in the accumulator before requesting data stream purchase. The properties of such an accumulator are as follows:

Generating an Accumulator: A secret key k_{Acc} is generated. The accumulator generation function $G_{Acc}()$ takes k_{Acc} and the revocation list L as inputs ad returns an accumulator U. The function is $U \leftarrow G_{Acc}(k_{Acc}, L)$

Updating an Accumulator: To add a new value v_{new} to the accumulator, function Upd_{Acc} takes current accumulator U, k_{Acc} and v_{new} to output updated accumulator $U_{new} \leftarrow Upd_{Acc}(U, k_{Acc}, v_{new})$

Generation of a Non-membership Witness: The non-membership witness generation function G_w is implemented as $w \leftarrow G_w(U, k_{Acc}, L, a)$ where a is a value not in L.

Verification of a Non-membership Witness: The non-membership witness verification function $V_w()$ is implemented as $0/1 \leftarrow V_w(U, w, a)$. It returns 0 if a is in L, else 1.

Usage-Controlled Access: Usage Control (UCON) (i.e., control the 'intended purpose' or 'expected use' of data) is an extension to traditional forms of access control such as RBAC or ABAC. It is widely used in data control technologies to implement data sovereignty and control data flow between applications (Bargh et al., 2018). While traditional access control mechanisms only authorize the user once before access is allowed (pre), usage-controlled access systems monitor the usage of resource and user's attributes on a continuous (ongoing) basis. Such an access control is instrumental for applications that are relatively long-lived and where immediate revocation is required, features required for an real-time IoT data service. Usage Control defines two types of attributes for both *User* and *Resource*: *mutable* and *immutable* attributes. Mutable attributes are user/resource attributes that undergo change as a result of access and immutable attributes do not change during the course of access. For e.g., a $buyer's$ prepaid balance has to be decreased as it consumes or access the data stream hosted on a *storage operator* node and hence its is a mutable attribute. $PDID$ is an immutable attribute in the this context. UCON monitors the *mutable* attributes during access to evaluate policies on an ongoing basis.

Usage-Controlled Access defines policies using six main elements namely: *User Attributes* (UA), *Resource Attribute* (RA), *Rights* (R), *Authorization (A), Obligation (OB)* and *Condition (C)*. In the proposed work, user and resource are represented by *buyer* and *data streams* respectively. UA and RA are sets of attributes that uniquely identify a user or resource. UA and RA are properties evaluated during the access decision process (Park and Sandhu, 2004). $Rights$ enables access to requested resources in a particular mode, such as read/download or perform computation on data. In the proposed work, *A, OB* and *C* are the three decision factors as opposed to traditional access models that only consider Authorization (Park and Sandhu, 2004). *Authorization* is based on UA and RA and associated $rights$. In the proposed work, authorization is based on *buyer* and data stream attributes and the corresponding defined rights on those data streams. UA is expressed as presentation of a valid VC issued by the SL of the requested data stream. *Obligations* are requirements that the *buyer* must perform before (pre) and during (ongoing) access. In the proposed work, pre-Obligation would be pre-payment and proving non-membership of revocation list. Ongoing-Obligation would be evaluated on an ongoing basis until the completion of data access and it includes rechecking revocation status on an ongoing basis and/or decreasing the $buyer's$ prepaid balance. And finally, *Condition* are system-oriented decision factors, hence these will be based on factors such as system load and IoT data stream real-time generation.

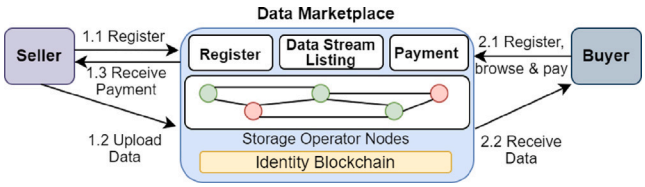

Fig. 3. System Actors in the Proposed Protocol.

4. Proposed protocol

4.1. Overview

A privacy-aware, authentication and usage-controlled access protocol for IIoT data marketplace is presented in this section. The primary actors in this protocol are the $Seller(SL)$, the *Buyer* and the $StorageOperator(SO)$ nodes as shown in Fig. 3. These system actors in our proposed protocol can be mapped to the VC ecosystem actors presented in the previous section. In this work, SL acts as an *issuer* as it issues VC to both SO and *buyer* signed by its private key K^{pr}_{SL}. SO and *Buyer* both are *holders* as they collect VC from SL. Since SO can verify any verifiable presentations from any holder, containing proofs of claims from any issuer, consequently it is a $verifier$ (Sporny et al., 2022). However, each *buyer* may or may not have that capability and it will depend if it can act as a *verifier* for each type of verifiable presentation from every issuer. In the proposed protocol, we assume that both SO and *buyer* will act as verifiers and will perform mutual authentication based on issued VC from SL.

The data marketplace provides a range of services including data listing, registration, payment for data, etc. Identity Blockchain provides a decentralized identity infrastructure to log registration of actors and issuance of VCs. The phases in the proposed protocol are *system setup and registration*, *authentication* and *UCON phase*. The high-level workflow is illustrated in Fig. 4. The system setup and registration phase is only performed once. In the setup phase, all the participants register themselves by creating DID and associated keys on the blockchain. After this step, the *Buyer* can browse the listed data streams on the marketplace advertised by several SL with their $rates$, purchase mode ($batchMode$ or $streamMode$) (Dixit et al., 2023) and $terms$. The purchase mode defines the frequency of payment made by the *Buyer*. In the registration phase, the *Buyer* interested in purchasing a data stream from SL presents a signed digital identity to the SL

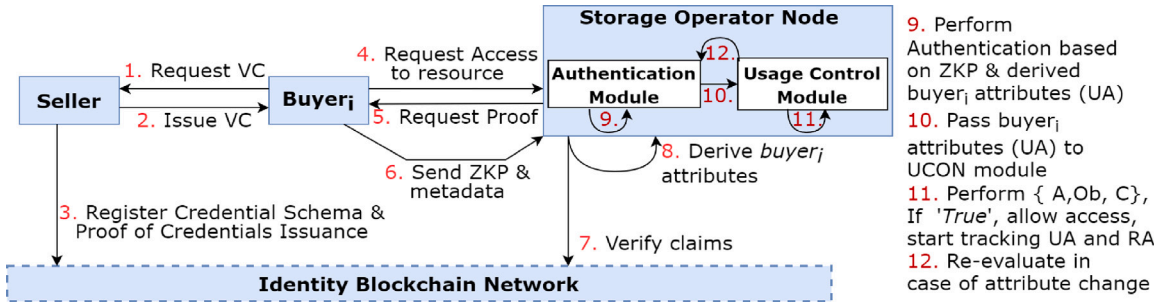

Fig. 4. Overview of proposed protocol with detailed flow of interactions between actors.

Table 3
Notations.

Symbol	Description
G	Generator point
DID_x	DID of x
$PDID_i$	Pseudo-identifier of $Buyer_i$
K_x^{pr}	Private key of x
K_x^{pu}	Public key of x
K_i	Secret key between SL and $Buyer_i$
K_{SOj}	Secret key between SL and SOj
$payment_{ID}$	Payment ID given to $Buyer$ after payment
$[VC_i]_{K_{SL}^{pr}}$	VC of $Buyer_i$ issued by SL
$[VC_{SO}]_{K_{SL}^{pr}}$	VC of SOj issued by SL
$[VC_{DigIDi}]_{K_{MC}^{pr}}$	VC of $Buyer_i$ issued by market consortium
$\oplus$	XOR Operation
$h(X)$	Hash of X
ϕ_{id}	$stream_{ID}$ of IoT data stream
w_i	Non-membership witness of revocation List L

for verification. If the digital identity verification along with requested payment is successful, the SL generates a VC with claims including the $payment_{ID}$. SL signs the VC with its private key and issues it to the $Buyer$. Similarly, SO_j also receives a VC from SL after identity verification. In the proposed protocol, EdDSA (Edwards-curve Digital Signature Algorithm) (Nakov, 2023) scheme based on elliptic curve cryptography (ECC) is used for VC signing and verification.

After registration phase, the $Buyer$ goes through authentication followed by usage-controlled access phase with the SO hosting the purchased data stream on behalf of SL. To authenticate, $Buyer$ sends the data access request to the SO. It then generates the ZKP of the VC and sends it to the SO. The SO_j verifies the received proof to confirm the $Buyer's$ right to access a data stream during the authentication part of the protocol. Thereafter, the usage-controlled access protocol evaluates the $Buyer's$ attributes presented in VC to authorize it. From thereon, the buyer and resource attributes are monitored on an ongoing bases until the data consumption is completed/terminated. The notations used in the proposed protocol are presented in Table 3.

4.2. System setup and registration phase

4.2.1. System setup

Step 1: The DID document corresponding to the SL's DID, DID_{SL}, is stored on the blockchain. SL generates a private key K_{SL}^{pr}. The public key is generated through EdDSA key generation function. Let G be the generator point. The public key K_{SL}^{pu} is a point on the elliptic curve, calculated by EC point multiplication as $K_{SL}^{pu} = K_{SL}^{pr} * G$. K_{SL}^{pu} is stored in the SL's DID document.

Step 2: $Buyer_i$ generates a DID, DID_i, and stores the corresponding DID document on the blockchain. Then, $Buyer_i$ generates a pair of private (K_i^{pr}) and public (K_i^{pu}) keys using the EdDSA key generation function similar as discussed in Step 1. $Buyer$ stores K_i^{pr} in its digital wallet. K_i^{pu} is stored in the $Buyer$'s DID document.

After this step, the $Buyer$ is onboarded on the data marketplace platform and can start browsing the listed data offers from various $sellers$. Once the $Buyer$ decides to purchase a data stream with ID ϕ_{id}, it enters into an agreement with its SL using the registration phase.

Step 3: Storage Operator, SO_j, also generates its DID (DID_{SOj}) and stores its public key immutably in its DID document on the blockchain.

Step 4: SL generates an accumulator, a revocation list L and a secret key k_{Acc}. Initially, L does not have any elements. As discussed in Section 3, SL generates an accumulator U from k_{Acc} and L using G_{Acc} function.

Step 5: SO_j hosts data streams with associated usage-controlled access policies as defined by SL in order to provide access to legitimate $Buyer$s only on providing a valid set of attributes.

4.2.2. Registration phase

in this phase, the registration of the $Buyer_i$ with SL on the marketplace takes place to initiate data trading. SO_j also register on the marketplace as it agrees to host data for SL.

Buyer Registration: consists of the following steps:

Step 1: $Buyer$ holds $[VC_{DigIDi}]_{K_{MC}^{pr}}$, the VC of their digital identity issued by a trusted party (e.g., marketplace consortium) after KYC verification. The process is very similar to an e-Know Your Customer (eKYC) process done by an organization to onboard customers. Next, $Buyer_i$ generates a pseudo-identity $PDID_i$. The $buyer$ can generate a new pseudo-identity for trading with each new seller. In this way the activity of $buyer$ on the marketplace cannot be tracked and linked. $Buyer$ decides to purchase the data stream with ID ϕ_{id} using mode $md(batch/stream)$ from SL. If the mode selected is $batch$, $Buyer$ will need to pay only once for a burst mode of data, whereas in $stream$ mode, payment needs to be made periodically to receive real-time IoT data (Dixit et al., 2023). Then, the buyer generates a message M_{REG1} with a registration request, streamID ϕ_{id}, mode md, DID_i, $PDID_i$ and $[VC_{DigIDi}]_{K_{MC}^{pr}}$ and sends it to SL.

Step 2: SL verifies the $Buyer's$ digital identity using the EdDSA's signature verification. After that, SL generates a nonce n and a key K_i for future communication with $Buyer_i$. A nonce is a random number used only once to provide various types of security guarantees. At this stage, the nonce n prevents against reflection attack as explained later. Then, SL generates a payment request Pay_{req} for requested stream ID ϕ_{id} to be consumed in the mode md. Then, the payment request, K_i and n are concatenated and its hash value hV is computed. After that, the SL generates a message M_{REG2} with hV, Pay_{req}, K_i and n and sends it to the buyer.

Step 3: $Buyer_i$ receives M_{REG2} from SL. $Buyer_i$ makes payment on the marketplace to receive $payment_{ID}$ for this transaction. Then $Buyer_i$ generates a nonce N_i. The nonce N_i will prevent the replay attack by a malicious entity as explained later. Then, $payment_{ID}, K_i$, n and N_i are concatenated and its hash value $hV2$ is computed. Then, $Buyer_i$ generates a message M_{REG3} with $hV2, payment_{ID}$, and N_i and sends it to the SL.

Step 4: After SL has successfully verified the payment details, it generates a nonce n^{new} and a credential $cred$. The newly generated

Algorithm 1: *Buyer* REGISTRATION PHASE

1: $Buyer$:
Generate: DID_i, $PDID_i$
Generate: $K_i{}^{pr}$, $K_i{}^{pu}$ pair
Select: data stream ID (ϕ_{id}) & $md(stream/batch)$
M_{REG1} :
$\{Req_{id}, \phi_{id}, md, [VC_{DigIDi}]_{K_{MC}^{pr}}, DID_i, PDID_i\}$
$Buyer \rightarrow SL$: M_{REG1}
2: SL:
$Verify_{EdDSA}$: $[VC_{DigIDi}]_{K_{MC}^{pr}}$
Generate: K_i, nonce n
Generate: Payment Req for (ϕ_{id}) in mode md
hV = h ($Pay_{req} \| K_i \| n$)
M_{REG2} : $\{hV, Pay_{req}, K_i, n\}$
$SL \rightarrow Buyer$: M_{REG2}
3: $Buyer$ makes payment and receives $payment_{ID}$
Generate: N_i
$hV2$ = h ($payment_{ID} \| K_i \| n \| N_i$)
M_{REG3} : $\{hV2, payment_{ID}, N_i\}$
$Buyer \rightarrow SL$: M_{REG3}
4: SL:
Generate: n^{new}, credential $cred$
hV_{cred} = h ($cred \| K_i^{pu} \| n^{new}$)
Sign hV_{cred} with K_{SL}^{pr}
i.e., $[VC_i]_{K_{SL}^{pr}} = Sign_{EdDSA}(hV_{cred})$
$n^{*new} = K_i \oplus n^{new}$
$w_i = G_w(U, k_{Acc}, L, PDID_i)$
M_{REG4} : $\{[VC_i]_{K_{SL}^{pr}}, cred, n^{*new}, w_i\}$
Store: $DID_i, PDID_i, n^{*new}, w_i, K_i$
$SL \rightarrow Buyer$: M_{REG4}
5: $Buyer$: **Store:** $[VC_i]_{K_{SL}^{pr}}, cred, K_i, w_i n^{*new}$

nonce n^{new} will act as a shared secret between buyer and seller for the next round of interaction to enhance security as explained later. The generated credential, $cred$, is a set of claims that the $Buyer_i$ can assert by presenting it to the right verifier. After that, $cred, K_i^{pu}$ and n^{new} are concatenated and its hash value hV_{cred} is computed. Then the SL signs the credential with its private key using EdDSA's signature generation algorithm.

For that the SL generates a secret integer r and a random point is generated as:

$$r = hash(hash(K_{pr}^{SL}) + m) mod\ q; R = r * G \tag{1}$$

Then the SL calculates the signature as:

$$h = hash(R + K_{pu}^{SL} + m) mod\ q \tag{2}$$

$$s = (r + h * K_{pr}^{SL}) mod\ q \tag{3}$$

The VC, $[VC_i]_{K_{SL}^{pr}}$ of $Buyer_i$ indicates that the $Buyer_i$ is a valid registered user who has made payment for requested data stream. It can be verified using the corresponding public key K_{pu}^{SL} of the SL. The SL also generates a non-membership witness w_i for $Buyer_i$ by calling $G_w()$ function with inputs U, k_{Acc}, L, and $PDID_i$ as discussed in Section 3. The new nonce n^{new} is XORed with K_i to produce n^{*new}. This operation ensures that the shared secret n^{new} is only decipherable by the intended recipient with the shared secret key K_i i.e. $buyer_i$. The SL stores DID_i, $PDID_i$, n^{new} and K_i for future communications with the $Buyer_i$. After that, SL generates a message M_{REG4} with $[VC_i]_{K_{SL}^{pr}}, cred, n^{*new}$ and w_i and sends it to the buyer.

Step 5: $Buyer_i$ receives M_{REG4} from SL. It stores $[VC_i]_{K_{SL}^{pr}}$, $cred, K_i, w_i$ and n^{new} in its digital wallet.

SO Registration: The steps are as follows:

Step 1: Similar to the *Buyer* registration, SOj registers with SL and receives a VC, VC_{SOj}, signed by SL with K_{SP}^{pr}. However, it skips the steps for payment as done by the buyer. It sends a request for VC to the SL to attest to its legitimacy of hosting data streams for that SL. It stores received VC_{SOj}.

Step 2: SL produces key K_{SOj} and sends it to SO_j. SO stores K_{SOj} for future communication with SL.

4.3. Authentication and UCON phase

4.3.1. Authentication phase

$Buyer_i$ must be authenticated by SO_j to access the data stream hosted by SO_j on its storage network on behalf of SL. $Buyer_i$ must also authenticate SO_j.

Step 1: after successful registration with SL and receiving valid VC that will help it to access the data stream it purchased on the marketplace, $Buyer_i$ authenticates with SO_j. $Buyer_i$ generates M_{A1} with $PDID_i$, requests to access stream ϕ_{id} and requests for the SO's VC. Then, M_{A1} is sent to SO_j. $Usage_{req}$ in M_{A1} represents the request of the buyer to access a data stream with stream ID ϕ_{id} from SO.

Step 2: SO_j composes and sends a message M_{A2} to $Buyer_i$ with $DID_{SOj}, [VC_{SOj}]_{K_{SP}^{pr}}$ and a request of proof of $Buyer$'s VC to validate its claim to access the data stream.

Step 3: $Buyer_i$ receives M_{A2} from SO_j. Then, $Buyer_i$ verifies SO_j's VC by using the EdDSA signature verification algorithm using K_{SP}^{pu}. After that, credential $cred$, K_i and nonce n^{new} are concatenated and the resultant hash value is computed and stored. Next, the $Buyer_i$ produces a ZKP of the VC using the $Prove$ algorithm with an output π as mentioned in Table 2. This is the ZKP of the VC and $Buyer_i$ presents it to SO_j to prove in Zero Knowledge that it holds a valid VC. Then, $Buyer_i$ generates M_{A3} using π and w_i and sends it to the SO_j.

Step 4: when SO_j receives M_{A3}, it verifies π using the $Verify$ algorithm as mentioned in Table 2. The SO can only know that $Buyer_i$ holds a valid credential and some metadata about the data stream. It cannot learn anything about its identity. Then the SO_j verifies the non-membership witness w_i of the $Buyer$ by calling V_w function as explained in Section 3 to confirm that the buyer is not revoked during this session. Once the $Buyer_i$ and SO_j are mutually authenticated and the SO is assured of the buyer's claims and revocation status, it passes the $Buyer's$ metadata to the policy evaluation engine that manages access of data streams on the network.

It is worth noting that during the complete process of registration and authentication, all the communications are P2P between the interacting parties i.e., there is no centralized party involved to acquire keys to authenticate or authorize users. It enables the decoupling of issuers and verifiers and they can function independently. For example, if for any reason the issuer goes offline, the verifier can still verify the authenticity and claims of the buyer.

4.3.2. Usage-controlled access policy evaluation

This evaluation takes place at SO_j node. After the authentication is completed, the $buyer's$ metadata is passed to the policy evaluation engine in the UCON module. Policy Engine parses the metadata ($PDID_i$, mode of access, ϕ_{id} etc.) received from the $buyer's$ VC to evaluate the

Algorithm 2: AUTHENTICATION PHASE

1: $Buyer \rightarrow SO$: $MA1$: $\{PDID_i, Usage_{req}, VC_{req}\}$
2: $SO \rightarrow Buyer$: $MA2$
$MA2$: $\{DID_{SOj}, [VC_{SOj}]_{K_{SP}^{pr}}, Proof_{req}\}$
3: $Buyer$:
$Verify_{EdDSA}$: $[VC_{SOj}]_{K_{SP}^{pr}}$
Calculate: hV_{cred} = h ($cred \| K_i^{pu} \| n^{new}$)
Store: hV_{cred}
$\pi = Prove([VC_i]_{K_{SP}^{pr}})$
$MA3$: $\{\pi, PDID_i, w_i\}$
$Buyer \rightarrow SO$: $MA3$
4: SO : **Verify**: π
$V_w(U, w_i, PDID_i)$ = ?1

policy against its attributes. The usage-controlled access components are presented in Table 4. As we can observe, the metadata along with the verified VC make up the $User\ Attributes$ set. The resource in this case is the data stream that would be accessed by the buyer hence; $Resource\ Attributes$ can be represented as a set of attributes like the seller ID who owns this data stream(SL_{ID}), name of the stream, time of upload/creation, etc. $Rights$ defines how the resources can be used by the user and in this case they are represented by the mode of access and whether they can be read, downloaded or computed on. Pre-obligation ($preOB$) checks are defined and they define that a user must provide a non-membership proof and must have completed the payment and provide a paymentID as proof. Similarly, Ongoing-Obligations ($onOB$) define that during ongoing access, the non-membership proof must hold true along with allotted time and if pre-paid balance and stream mode are opted it should be available to charge payments.

Since pre-payment is completed, $Buyer_i$ is verified and since it has provided a witness of non-membership (w_i) of the revocation list, it passes the authorization and pre-obligation phase. The data usage begins after this point and hence the ongoing obligation check begins. The UCON module continuously monitors the $Buyer's$ (UA) and Data stream's (RA) attributes to record any change against defined ongoing obligations. Access is terminated based on the termination of the allotted access period or in case of any of the ongoing obligation violations, whichever happens first. In short, a Usage Policy is defined as $Policy = \{UA \wedge RA \wedge Right\}_{A,OB,C}$ i.e., evaluate policy using elements of set UA, RA and R under defined A, OB (both pre and on) and C. The *allow(UA, RA, R)* predicate indicates that the user with attributes UA is allowed to access the resource with attributes RA with rights R only if the indicated $preOB$ conditions are fulfilled. The *Fulfilled(preOB)* predicate tells us if each of the required obligations is true to start the access. It is worth noting that the UCON policy evaluation takes place before the start of every new access request by a *buyer* to the *storageOperator* node and each session is continuously monitored until it is terminated or completed. The usage-controlled access components defined in Table 4 stand true for the proposed protocol and it is a business-logic dependent implementation, hence it can vary with various implementations.

5. Implementation and result evaluation

5.1. System components

(1) *Hyperledger Indy:* the proposed framework is blockchain platform agnostic as long as it supports the digital identities rooted in the blockchain. Hyperledger Indy (henceforth called Indy), an open-source project of the Linux Hyperledger foundation, provides a decentralized identity implementation including support for a ledger, DID' s implementation and all features related to VCs. It supports the building of applications that interact with the ledger for decentralized identity management. It is an open source, identity blockchain platform that permits pluggable components for a wallet (also referred to as an agent), DID resolver and Indy nodes among other cryptographic modules.

(2) *System Actors and Roles:* the system includes the agents (henceforth called '*wallets*') of each subject i.e., SL, SO and $Buyer$. The agents are represented as separate docker containers for each actor. In the proposed case, the seller acts as the issuer of VCs to both the buyer and storage operator, while the buyer and storage operators both perform mutual verification of credentials before data exchange. The *authentication module* and *usage control module* within the Storage operator node as shown in Fig. 4 are emulated as services within the same docker container.

(3) *Interaction Details:* During system setup, each actor SL, $Buyer_i$ and SO_j create a DID and associated key pair using their Indy wallets and register themselves on the blockchain. This only requires one call for each wallet to the ledger. Later in order for the SL to issue VC to the $Buyer_i$ and SO_j, it has to perform some setup. Before SL can issue VC, it requires completing two VC setup steps after registration on the blockchain: VC Schema generation and VC Definition generation. VC Schema generation involves the creation of the schema by SL and sending it to the ledger. VC definition generation is completed after two interactions with the ledger, first to get the schema of the VC and later to store the generated VC definition on the ledger. Now VC creation for issuing can take place and this comprises two calls to the ledger from the issuer's (SL) agent.

5.2. Evaluation

The experiment was set up on Google Cloud Computing Platform (GCP) running an instance of Linux Ubuntu configured with 16vCPUs, 64 GB RAM and 100 GB HDD storage. The experiments were performed by varying the number of parameters to understand the dependency on each of them. The parameters used are the number of blockchain nodes i.e. (N), the number of claims (C) in a VC issued to a buyer and the number of $obligations$ (both pre and ongoing) imposed in the access policy check. We test the setup by varying the number of claims in each VC and later by varying the number of blockchain nodes in the Indy ledger. The wallets are written in Python 3.2. In order to perform usage-based access control, a cronJob scheduler (Lewis and Drake, 2021) was written to interact with the underlying Unix operating system. For VC signing and verification, we use EdDSA, particularly Ed25519 (Nakov, 2023), which is based on a performance-optimized elliptic curve, Curve25519. Cryptosystems based on ECC have the advantage of using small key and signature sizes. Hence, they have low memory and computational resource requirements. For the ZKP system, we used the API implementation given in Salleras (2023). The implementation of this protocol is available online on Github (Dixit and Jadon, 2022). In order to understand the performance of the protocol, we perform tests on each stage to see how the system performs with variations in the above-discussed parameters.

System Setup and Registration: Fig. 5 presents the system setup time which can be divided into three steps: issuer (SL) registration, VC schema generation and VC definition generation. These individual setup time and combined setup time for each claim and node can be observed in this figure.

It can be observed that the time taken for SL (issuer) registration remains approximately the same with an increase in the number of claims (C) for a specific number of nodes; however, with an increase in the number of nodes (N), the registration time increases. This is due to the fact that with an increase in the number of blockchain nodes, the transaction synchronization time among nodes increases. This observed pattern is due to the fact that the real bottleneck here is the transaction processing, and not the consensus protocol itself as elaborately explained in the work (Thakkar and Natarajan, 2021). The transaction synchronization time can be reduced by vertical scaling of the system, i.e., increasing the processing power of each node. A similar pattern is observed with time taken for VC Schema generation and VC definition generation. It is worth noting that since the system setup is a one-time process, it is a feasible overhead and with vertical scaling performance can be improved. Here, it is important to note that, N represents the number of Indy nodes which is part of the decentralized identity infrastructure and it does not represent other blockchain nodes that may be employed to perform business logic in a data marketplace (Dixit et al., 2023).

Once the system is set up and the issuer (SL) is ready to issue credentials (VC), we can look at the time taken for VC creation and henceforth verification. As shown in Fig. 6, the time for VC creation (C) and verification (V) increases with no. of claims. It also rises with the number of nodes for the same reason as described above. However, even with the number of claims being as high as 20, the value remains considerably low, particularly for VC verification. It is to be noted that

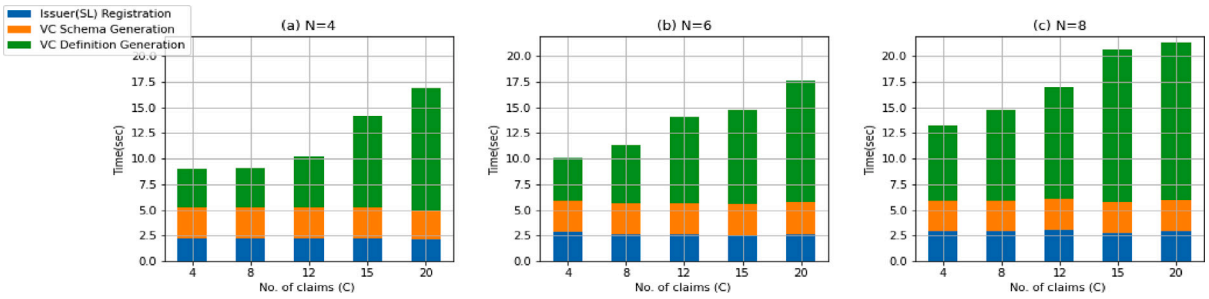

Fig. 5. System Setup Time with varying number of Indy Nodes (N) and number of claims (C).

Table 4
Usage-controlled access components.

Elements	Explanation
User/Buyer Attribute(UA)	$PDID_i$, $payment_{ID}$ $prepaidBalance$, duration=X
Resource/Data Stream Attribute(RA)	SL_{ID}, ϕ_{ID}, $streamName$, Time($DDMMYY$)
Rights(R)	$mode$(batch/stream) (Read, Download, Compute)
Pre-Obligation(preOB)	w_i, $payment_{ID}$! = 0
Ongoing-Obligation(onOB)	w_i, $allotedTime > 0$ & $prepaidBalance > 0$
UCON Policy (λ)	$allow\ (UA, RA, R) \Rightarrow \lambda\ (UA, RA, R)$ $\wedge\ Fulfilled\ (preOB)$

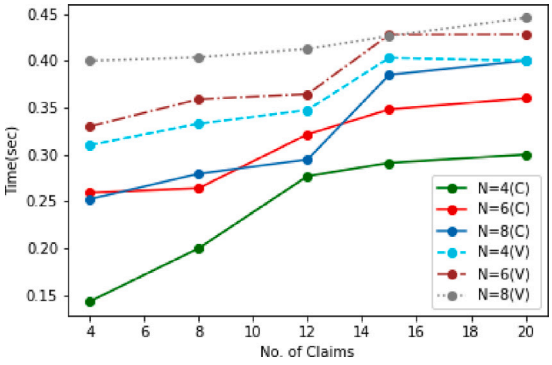

Fig. 6. Time taken for VC Creation/Verification.

Table 5
Storage Cost: $SL/Buyer$ Wallet [C = No. of claims].

For SL After	C=4	C=8	C=12	C=15	C=20
Wallet Creation(B)	258	258	258	258	258
VC Schema Gen.(B)	580	589	633	633	640
VC Definition Gen.(KB)	6.3	8.2	10.7	12.5	15.6
VC creation(KB)	15.9	20.2	26.2	30.4	34.4
For *Buyer* After					
Wallet Creation(B)	261	230	230	230	230
VC Receive(KB)	22.4	28.4	36.7	42.8	51.7

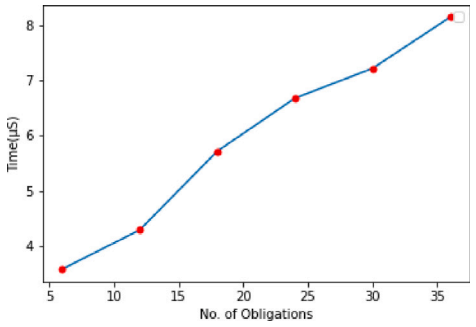

Fig. 7. Time taken for usage-controlled policy evaluation.

Table 6
Computation cost independent of VC setup.

Phase	Buyer	SL/SO
Register	–	$Verify_{EdDSA}$+ $Sign_{EdDSA}$ = 24.98 ms
Authenticate	$Verify_{EdDSA}$+ $Prove_{ZKP}$ = 18.27 ms	$Verify_{ZKP}$ = 1.1 ms
$Sign_{EdDSA}$ = 10.28 ms; $Verify_{EdDSA}$ = 14.7 ms		
$Prove_{ZKP}$ = 3.57 ms; $Verify_{ZKP}$ = 1.1 ms		

while VC issuance happens once for that VC, verification will happen multiple times so it is an important parameter to understand the system speed.

In order to understand the storage cost of the SL and $Buyer_i$ wallets as the VCs are issued, we can observe the values in Table 5. The table represents the values of the wallet storage cost of seller and buyer at each stage of VC computation. We observe that the storage cost for SL after VC Schema generation and VC definition increases steadily with an increase in the number of claims however it still remains in the order of bytes and KB, respectively. Similarly, the wallet storage cost for SL increases with the number of claims increase and remains in the order of KB. The storage cost for the *buyer* on receiving VC grows very reasonably with the number of atomic claims in VC. Even with 20 claims, the storage cost remains in the order of KB. A small variation in the $buyer's$ wallet creation cost can be seen in the first two values as it was averaged over a run of multiple trails and does not indicate any significant deviation in the result.

Authentication and Usage-Controlled access phase: While the setup and registration process are executed only once, the authentication and usage-controlled phase will be executed each time the $Buyer_i$ establishes a new session with SO_j. The total Computation Cost for the authentication phase as calculated during the implementation is shown in Table 6. While the time taken for VC computation during registration is dependent on the number of claims in VC generation, others are constant and depend on the type of operations. Since the time taken by XOR and concatenation operation is negligible, they are not considered in evaluating computation cost during authentication. The $Buyer_i$ incurs negligible cost during the registration phase and hence not shown in this table. During the authentication, the $Buyer_i$'s device executes $Verify_{EdDSA} + Prove_{ZKP}$ operations, which equals to 18.27 ms, while the SO executes operation $Verify_{EdDSA}$+$Sign_{EdDSA}$ during the registration phase, which equals to 24.96 ms. For verification of presented VC, the verifying party (SL/SO) performs the operation $Verify_{ZKP}$ and it only incurs 1.1 ms of time.

```
role Sx{
fresh  n, ki, payreq, nnew, cred, VCi, wi : Nonce ;
var DIDb, PDIDb, md, phiid, reqid,  Ni, payid  : Nonce;

recv_1(Bi,Sx, {Bi, DIDb, PDIDb, md, phiid, reqid}
pk(Sx) );
fresh hV: Response;
send_2(Sx, Bi, {{DIDb, n, ki} sk(Sx), payreq, hV} pk(Bi),
hash(hV) );
var hV2: Response;
recv_3(Bi, Sx, {{ki, Ni} sk(Bi), hV2, payid } pk(Sx),
hash(hV2)  );
send_4(Sx, Bi, { {VCi} sk(Sx), cred, xor(ki, nnew), wi }
pk(Bi) );

claim(Sx, Running, Bi, nnew, ki);
claim_S1(Sx, Alive);
claim_S2(Sx, Weakagree);
claim_S3(Sx, Secret, Ni);
claim_S4(Sx, Niagree);
claim_S5(Sx, Nisynch);}
```

Fig. 8. Role for Seller Sx in SPDL.

Fig. 7 illustrates the computation cost incurred during the usage-controlled access phase as a function of a number of obligations. Fig. 7 was plotted by observing the time taken for resource consumption of a constant file size with varying numbers of ongoing obligations to check during the access phase. The computation cost observed during authentication in Table 6 and time taken for usage-policy evaluation in Fig. 7 are in the order of ms and μs and hence feasible in a real-world use case with further operational optimization.

6. Security analysis

In this section, we thoroughly analyse the proposed protocol in terms of security. We present a formal security verification using the broadly accepted automated software validation tool, known as Scyther tool (Cremers, 2008). We also carry out security analysis by intuitive reasoning through non-mathematical (heuristic) approaches to discuss the most common security threats and vulnerabilities in authentication and access management protocols and how the proposed work overcomes them in a decentralized setting such as a data marketplace.

6.1. Formal security verification: A simulation study using Scyther tool

In this section, the proposed protocol's resilience to different attack vectors in authentication protocol is assessed using the widely accepted automated software validation tool, Scyther tool. Through the simulation study using this tool, we show that the proposed scheme is safe against attacks such as impersonation, passive secret disclosure, man-in-the-middle and replay attacks and hence provides strong authentication.

Scyther (Cremers, 2008) is a security tool that can be used for verification, falsification and analysis of security protocols. It is based on a pattern refinement algorithm, providing concise representations of (infinite) sets of traces. The Scyther framework allows to model adversaries in security protocol analysis, varying between Dolev–Yao style adversaries to other more powerful adversaries. Also, it supports security notions such as key compromise, impersonation, weak perfect forward secrecy and adversaries capable of state-reveal queries (Cremers, 2008). The protocol to be verified is provided to the Scyther tool in the form of a protocol description written using the Security Protocol Description Language (SPDL). The protocol description comprises a set of roles, with each role consisting of a sequence of events. The events can be sent or received in terms (security parameters).

The actors *Seller* (Sx), *Buyer* (Bi) and *StorageOperator* (So) are communicating with one another in the proposed protocol. They are modelled as roles (Sx), (Bi) and (So) as shown in Figs. 8, 9 and 10,

```
role Bi
{
fresh DIDb,PDIDb,md,phiid,reqid,Ni,payid,PDIDi,usagereq,
vcreq,wi: Nonce;
var payreq,n,ki,cred,nnew,VCi,DIDso,VCso,proofreq: Nonce;

send_1(Bi,Sx,{Bi, DIDb, PDIDb, md, phiid,reqid} pk(Sx));

var hV: Response;
recv_2(Sx,Bi,{{DIDb, n, ki}sk(Sx),payreq, hV}pk(Bi),
hash(hV));

fresh hV2: Response;
send_3(Bi, Sx, {{ki, Ni} sk(Bi), hV2, payid} pk(Sx),
hash(hV2) );
recv_4(Sx,Bi,{{VCi}sk(Sx),cred, xor(ki, nnew),wi}pk(Bi));
send_5(Bi,So,{Bi, DIDb, usagereq, vcreq}pk(So));
recv_6(So,Bi,{{DIDb, DIDso, VCso}sk(So),proofreq}
pk(Bi));
fresh pi, hVcred: Response;
send_7(Bi,So,{{pi, DIDso,PDIDi,hVcred }sk(Bi),wi}pk(So),
hVcred);

claim(Bi, Running, Sx, nnew, ki);
claim_B1(Bi, Alive);
claim_B2(Bi, Weakagree);
claim_B3(Bi, Secret, ki);
claim_B4(Bi, Secret, n);
claim_B5(Bi, Niagree);
claim_B6(Bi, Nisynch);}
```

Fig. 9. Role for Buyer Bi in SPDL.

```
role So{
fresh DIDso,proofreq,VCso: Nonce ;
var DIDb,PDIDi,usagereq,vcreq,wi: Nonce;

recv_5(Bi, So, {Bi, DIDb, usagereq, vcreq} pk(So) );
send_6(So, Bi, {{DIDb, DIDso, VCso} sk(So), proofreq}
pk(Bi) );
var pi, hVcred: Response;
recv_7(Bi, So,{{pi, DIDso, PDIDi,hVcred}sk(Bi),wi}
pk(So),hVcred) ;

claim_So1(So, Alive);
claim_So2(So, Weakagree);
claim_So3(So, Secret, pi);
claim_So4(So, Niagree);
claim_So5(So, Nisynch);}
```

Fig. 10. Role for Storage Operator So in SPDL.

respectively. A role begins with the declaration of the sending and receiving terms, then the exchange of such terms followed by the security claims. The security claims are used to model the protocol's security properties. These claims are a crucial part of the protocol design without which Scyther tool would not know what is to be verified. The security claims verified in this work using Scyther tool are (i) aliveness (ii) weak agreement (iii) non-injective agreement (Niagree) and (iv) non-injective synchronization (Nisynch). Protocols satisfying *aliveness* guarantee that the communicating party has sent a message in the past. A protocol guarantees to an agent a in role A *weak agreement* of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then b has been running the protocol believing to be communicating with a. The claim *Weak agreement* is essential to check if the authentication is successful. Similarly, a protocol guarantees to an agent a in role A *non-injective agreement* of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then b has been running the protocol believing to be communicating with a and a and b agree on the contents of all the messages exchanged. And finally, a protocol guarantees *non-injective synchronization* if an agent a in role A completes a run as the initiator with b as the responder, then b has run the protocol as responder with a, all messages sent and received are exactly as described in the protocol

Table 7
Scyther results: Verification of reachability of the roles for Sx, Bi and So.

Scyther results: characterize				
Claim			Status	
Proposed	Sx	Reachable	Ok	Verified
	Bi	Reachable	Ok	Verified
	So	Reachable	Ok	Verified

specifications, in same order. The two claims *non-injective agreement* and *non-injective synchronization* are used to verify if our protocol is protected against replay and man-in-the-middle attacks.

The output presented in Table 7 confirms that the roles Sx, Bi and So are reachable. This ensures that non-reachability does not lead to any obvious weakness in the protocol description. The verification results in Table 8 give useful insights into claims made on roles. Our claims on role Sx include (i) aliveness, (ii) weak agreement, (iii) secrecy of nonce Ni, (iv) non-injective agreement and (v) non-injective synchronization. Secondly, our claims on role Bi comprise (i) aliveness, (ii) weak agreement, (iii) secrecy of key ki, (iv) secrecy of nonce n, (v) non-injective agreement and (vi) non-injective synchronization. Finally, our claims on role So are (i) aliveness, (ii) weak agreement, (iii) secrecy of proof pi, (iv) non-injective agreement and (v) non-injective synchronization. Firstly, the protocol guarantees the secrecy of nonce n and secret key ki generated by seller to establish secure communication with the buyer to complete its registration. Similarly, the secrecy of nonce Ni is assured hence verifying the transmission of payment details from buyer to seller is guaranteed. Since similar steps are used for the storage operator registration, hence, secrecy of similar protocol elements between SL and SO is assured. With these guarantees, impersonation and passive secret disclosure attacks are prevented. The claims (Sx, Running, Bi, nnew, ki) and (Bi, Running, nnew, ki) defined under the roles of Sx and Bi are used in order to specify data agreement between Sx and Bi. Other security claims ensure authentication success and protection against replay and man-in-the-middle attacks.

6.2. Informal security analysis

In this section, we show that the proposed protocol is secure against various attack vectors by intuitive reasoning. It is worth noticing that we follow the informal (non-mathematical heuristic) security analysis to show the proposed protocol is secure against other attacks that are not all covered in the previous subsection.

Credential Theft/Stuffing: Attackers often steal credentials to gain unauthorized access in traditional centralized systems. They gain access to authentication servers to steal user credentials (Fruhlinger, 2023). This is possible as identifiers and credentials of users are stored in a single verifying repository. In the proposed protocol, we leverage the concept of DIDs and VCs that enable decentralized identity and credential management. The lifecycle of DID is pinned in the underlying identity blockchain, which is essentially a decentralized database of these records. It is fault-tolerant, enables efficient revocation and is secure to credential stuffing. The *Buyer* creates and manages their own IDs and does not depend on the marketplace to maintain them. The users are empowered by the fact they have complete control of their IDs. While in the proposed protocol, the credentials are still susceptible to be stolen/leaked, by not having a centralized repository it becomes less attractive to attackers and at times not worthy of amount of the effort required.

Inference Attack: Inference attacks are possible by mining information available about patterns of a user on any platform. By drawing information from their digital footprint, user activity can be linked and tracked in a marketplace. This leads to loss of privacy and unsolicited marketing communications. In the proposed protocol, there are two safeguards against the possibility of an inference attack. Firstly, the *buyer* always interacts with other actors in the marketplace using a pseudo-ID $PDID_i$. For two successive sessions x and $x+1$, $PDID_i^x \neq PDID_i^{x+1}$. thus the identities of the users are unlinkable. Secondly, the *Buyer* provides a ZKP of the VC to the SO. The SO cannot learn anything about the $Buyer_i$ from the ZKP and only knows the claims against which policy evaluation has to be done. Thus, an adversary will be unable to link $Buyer's$ real identities with their pseudo-IDs or VC.

Table 8
Scyther results: verification of claims.

Scyther results: verify					
Claim				Status	Comments
datamarketplace	Sx	Proposed,S1	Alive	Ok	No attacks.
		Proposed,S2	Weakagree	Ok	No attacks.
		Proposed,S3	Secret Ni	Ok	No attacks.
		Proposed,S4	Niagree	Ok	No attacks.
		Proposed,S5	Nisynch	Ok	No attacks.

(a) Verifying Sx's claims

Scyther results: verify					
Claim				Status	Comments
datamarketplace	Bi	Proposed,B1	Alive	Ok	No attacks.
		Proposed,B2	Weakagree	Ok	No attacks.
		Proposed,B3	Secret ki	Ok	No attacks.
		Proposed,B4	Secret n	Ok	No attacks.
		Proposed,B5	Niagree	Ok	No attacks.
		Proposed,B6	Nisynch	Ok	No attacks.

(b) Verifying Bi's claims

Scyther results: verify					
Claim				Status	Comments
datamarketplace	So	Proposed,So1	Alive	Ok	No attacks.
		Proposed,So2	Weakagree	Ok	No attacks.
		Proposed,So3	Secret pi	Ok	No attacks.
		Proposed,So4	Niagree	Ok	No attacks.
		Proposed,So5	Nisynch	Ok	No attacks.

(c) Verifying So's claims

Non-Repudiation: After signing a statement with its private key, a party cannot deny having signed it (i.e., non-repudiation). The VC signing and verification are based on asymmetric cryptographic techniques. The SL signs the VC for the $Buyer_i$ and the SO_j with its private key (K_{SL}^{pr}). Only the SL knows the private key K_{SL}^{pr} to sign the VC. This guarantees non-repudiation.

Reflection Attack: A reflection attack occurs when an attacker tricks a legitimate user into performing an action by reflecting the user's request back to them. To prevent this attack, SL sends a nonce n in M_{REG2} : $\{hV, Pay_{req}, Ki, n\}$ to the *buyer* when requesting to make payment. Later, after completing the payment, the *buyer* uses the unique nonce n shared by SL to compute $hV2 = h(payment_{ID} \| K_i \| n \| N_i)$ in M_{REG2}.

Replay Attack: An adversary may intercept and retransmit messages exchanged between sender and receiver to get information (payment/data). To prevent it, during the *buyer* registration, when the buyer shares the completed payment details with SL, in M_{REG3}: $\{hV2, payment_{ID}, N_i\}$, a nonce N_i is included in the message to ensure that the message is unique. It will allow the SL to verify whether the nonce has been used before. The message is rejected if the nonce has already been used, preventing a malicious actor from registering with the SL.

Man-in-the-middle Attack: In order to prevent MITM attacks, several defence mechanisms have been provided in the protocol. Firstly, strong authentication guarantees are provided by the usage of VCs. The SO presents its VC to $Buyer_i$ which $Buyer_i$ verifies. Then, the *Buyer* presents the ZKP of its VC signed and issued by SL to the SO ad the SO verifies it. Only legitimate buyers and legitimate SOs hold the valid VCs, signed with the private key of the SL, K_{SL}^{pr}. Therefore, mutual authentication between actors makes it difficult for an impersonator to replicate the messages/requests. Secondly, since the sellers and buyers

in the network possess unique DID, secured by public–private key pair, an attacker cannot inject malicious signed messages.

Sybil Attack: In a Sybil attack, a malicious user creates multiple fake identities to gain control over a significant portion of the network. To prevent this attack, there are two security guarantees provided in the protocol. Firstly, on the SL level, before a buyer can register with a seller, it needs to present $[VC_{DigIDi}]_{K^{pr}_{MC}}$, the VC of their digital identity issued by the marketplace consortium after eKYC verification. This provides the first safeguard against the possibility of creating multiple fake buyer profiles on the marketplace. Secondly, on the SO level, a dynamic buyer revocation is proposed using a cryptographic accumulator. The marketplace maintains a revocation list and buyers must prove to SO before starting access that they are not members of this list by providing a non-membership witness issued by SL.

Privilege Abuse and Escalation: Excessive privilege in access control systems can lead to its abuse and escalated threats. Authorizing least-privilege access helps in putting a check on malicious actors who can harm the system if they get in maliciously. It ensures that users have only the required level of permissions, for the required duration and under required conditions in order to perform access. Usage-based access control performs ongoing along with pre-evaluation to determine the applicability of the right to access a certain resource. By limiting access under specific *authorization* policy governed by *obligations* and *conditions*, the attack surface (potential entry points for attackers) is reduced. In the proposed marketplace protocol, if any malicious actor gains access to a data stream hosted on a *storage operator* node, the checks in place on an ongoing basis will ensure to stop access effectively and efficiently.

7. Conclusion

In this work, we proposed a user-empowered, privacy-aware, authentication and usage-controlled access protocol for IIoT data marketplace. The proposed protocol addresses the issue of secure and privacy-aware authentication in real-time decentralized trading of IoT data, which is a future reality owing to the massive progress in advanced Artificial Intelligence algorithms. Along with presenting a secure and privacy-aware mutual authentication using DIDs and VCs, the protocol also implements usage-controlled access of data streams being accessed by the *buyer* from *storage operator*.

Payments on a data marketplace is a key step towards making it attractive for various parties and it can take various forms like traditional centralized payment methods or more recent decentralized payment techniques using crypto assets/NFTs (non-fungible tokens) etc. Cryptocurrency transactions when analysed using on-chain data have been found susceptible to traceability leading to deanonymization of the user and hence endangering their privacy (Fleder et al., 2015). Although some privacy-preserving cryptocurrencies like Monero (Kumar et al., 2017) and ZCash (Sasson et al., 2014) guarantee the privacy and untraceability of the user, broadly the use of payment channels can jeopardize the privacy of the user (Kumar et al., 2017). There are various approaches to preserve the privacy of the user on payment channels such as using mixers, user pseudonyms, privacy-preserving currency etc, however, such an analysis is out of the scope of this study but something to be aware of.

The proposed protocol is implemented to test its feasibility in terms of computation and storage performance. We observe that although the setup of such a protocol can be a bit expensive for the first time, its successive use for authentication and access control is computationally inexpensive. In addition, the security guarantees of using decentralized identifiers and credentials that can support privacy-preserving data trading support such a protocol.

CRediT authorship contribution statement

Akanksha Dixit: Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Bruno Bogaz Zarpelao:** Validation, Supervision, Methodology, Conceptualization. **Max Smith-Creasey:** Methodology, Conceptualization. **Muttukrishnan Rajarajan:** Supervision, Resources, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

Agbo, C., Mahmoud, Q., Eklund, J., 2019. Blockchain technology in healthcare: A systematic review. Healthcare (ISSN: 2227-9032) 7 (2), 56.

Banerjee, P., Govindarajan, C., Jayachandran, P., Ruj, S., Reliable, fair and decentralized marketplace for content sharing using blockchain. In: 2020 IEEE International Conference on Blockchain (Blockchain).

Bargh, M., Vink, M., Choenni, S., 2018. On using obligations for usage control in joining of datasets. In: Information Systems Security and Privacy. Springer International Publishing, Cham, pp. 173–196.

Camenisch, J., Lysyanskaya, A., 2002. Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Advances in Cryptology — CRYPTO 2002. pp. 61–76.

Cremers, C., 2008. The scyther tool: Verification, falsification, and analysis of security protocols. In: Computer Aided Verification. Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-540-70545-1, pp. 414–418.

Cui, J., Gu, N., Wu, C., 2022. Blockchain enabled data transmission for energy imbalance market. IEEE Trans. Sustain. Energy 13 (2).

Diego, S., Regueiro, C., Fernandez, G., 2021. Enabling identity for the IoT-as-a-service business model. IEEE Access 9, 159965–159975. http://dx.doi.org/10.1109/ACCESS.2021.3131012.

Dixit, A., Jadon, V., 2022. Supply chain. https://github.com/akankshadixit/SupplyChain/tree/main/docs.

Dixit, A., Singh, A., Rahulamathavan, Y., Rajarajan, M., 2023. FAST DATA: A fair, secure, and trusted decentralized IIoT data marketplace enabled by blockchain. IEEE Internet Things J. 10 (4), 2934–2944.

Figueredo, K., Seed, D., Wang, C., 2022. A scalable, standards-based approach for IoT data sharing and ecosystem monetization. IEEE Internet Things J. 9 (8), 5645–5652.

Fleder, M., Kester, M., Pillai, S., 2015. Bitcoin transaction graph analysis. arXiv: 1502.01657 [cs.CR].

Fruhlinger, J., 2023. Equifax data breach FAQ: What happened, who was affected, what was the impact?. [Online]: https://www.csoonline.com/article/567833/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html. (Accessed July 2023).

Groth, J., 2016. On the size of pairing-based non-interactive arguments. In: Advances in Cryptology – EUROCRYPT 2016. pp. 305–326.

Hasan, H., Salah, K., Yaqoob, I., Jayaraman, R., Pesic, S., Omar, M., 2022. Trustworthy IoT data streaming using blockchain and IPFS. IEEE Access 10, 17707–17721.

Jeedigunta, S., Hewett, N., Kaushik, P., Goel, A., 2023. Towards a data economy: An enabling framework. [Online]: https://www.weforum.org/whitepapers/towards-a-data-economy-an-enabling-framework. (Accessed July 2023).

Koch, K., Krenn, S., Marc, T., More, S., Ramacher, S., 2022. KRAKEN: A privacy-preserving data market for authentic data. In: Proceedings of the 1st International Workshop on Data Economy. DE '22, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450399234, pp. 15–20. http://dx.doi.org/10.1145/3565011.3569057.

Kortesniemi, Y., Lagutin, D., Elo, T., Fotiou, N., Nardone, R., 2019. Improving the privacy of IoT with decentralised identifiers (DIDs). J. Comput. Netw. Commun. (ISSN: 2090-7141) 2019.

Koutsos, V., Papadopoulos, D., Chatzopoulos, D., Tarkoma, S., Hui, P., 2020. Agora: A privacy-aware data marketplace. Cryptology ePrint Archive, Paper 2020/865. URL https://eprint.iacr.org/2020/865.

Kumar, A., Fischer, C., Tople, S., Saxena, P., 2017. A traceability analysis of Monero's blockchain. In: Foley, Simon N., Gollmann, Dieter, Snekkenes, Einar (Eds.), Computer Security – ESORICS 2017. Springer International Publishing, Cham, ISBN: 978-3-319-66399-9, pp. 153–173.

Kyriakidou, C., Papathanasiou, A., Polyzos, G., 2023. (Decentralized identity with applications to security and privacy for the internet of things). Computer Networks and Communications 1 (2).
Lewis, S., Drake, M., 2021. How to monitor cron jobs. [Online]: https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-ubuntu-1804. (Accessed July 2024).
Li, J., Li, N., Xue, R., 2007. Universal accumulators with efficient nonmembership proofs. In: Proceedings of the 5th International Conference on Applied Cryptography and Network Security. pp. 253–269.
Lim, J., Oh, H., Sim, K., Kim, S., Kim, K., 2023. A V2X access authorization mechanism based on decentralized ID (DID) and verifiable credentials (VC). In: 2023 International Conference on Information Networking. ICOIN, pp. 801–805.
Lu, J., Shen, J., Vijayakumar, P., Gupta, B., 2022. Blockchain-based secure data storage protocol for sensors in the industrial internet of things. IEEE Trans. Ind. Inform. 18 (8).
Ma, R., Zhang, L., Wu, Q., Mu, Y., Rezaeibagha, F., 2023. BE-TRDSS: Blockchain-enabled secure and efficient traceable-revocable data-sharing scheme in industrial internet of things. IEEE Trans. Ind. Inform. 1–10.
Nakov, S., 2023. "Eddsa and ed25519". [Online]: https://cryptobook.nakov.com/digital-signatures/eddsa-and-ed25519. (Accessed July 2023).
Oh, H., Park, S., Lee, G., Choi, J, Noh, S., 2020. Competitive data trading model with privacy valuation for multiple stakeholders in IoT data markets. IEEE Internet Things J. 7 (4).
Oku, R., Shiomoto, K., Ohba, Y., Decentralized identifier and access control based architecture for privacy-sensitive data distribution service. In: 2022 IEEE 8th World Forum on Internet of Things (WF-IoT).
Oracle, 2023. Oracle data marketplace. [Online]: https://docs.oracle.com/en/cloud/saas/data-cloud/data-cloud-help-center/AudienceDataMarketplace/AudienceDataMarketplace.html. (Accessed July 2023).
Park, J., Sandhu, R., 2004. The UCONABC usage control model. ACM Trans. Inf. Syst. Secur. (TISSEC).
Qi, S., Lu, Y., Zheng, Y., Li, Y., Chen, X., 2021. Cpds: Enabling compressed and private data sharing for industrial internet of things over blockchain. IEEE Trans. Ind. Inform. 17 (4).
Qi, M., Xu, Z., Wang, Z., Chen, S., Xiang, Y., Databox-based delivery service via blockchain. In: 2022 IEEE International Conference on Web Services.
Reyna, A., Martin, C., Chen, J., Soler, E., Diaz, M., 2018. On blockchain and its integration with IoT. Challenges and opportunities. Future Gener. Comput. Syst. (ISSN: 0167-739X) 88, 173–190.
Saidi, H., Labraoui, N., Ari, A., Maglaras, A., Emati, J., 2022. DSMAC: Privacy-aware decentralized self-management of data access control based on blockchain for health data. IEEE Access 10, 101011–101028.
Salleras, X., 2023. ZPiE: Zero-knowledge proofs in embedded systems. https://github.com/xevisalle/zpie. (Accessed July 2023).
Sarower, A., Hassan, M., 2023. Necessity of reliable self-sovereign identity management framework for resource constrained IoT devices. In: AIP Conference Proceedings, vol. 2579, (1), AIP Publishing.
Sasson, E. Ben, Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M., 2014. Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. http://dx.doi.org/10.1109/SP.2014.36.
Sporny, M., Longley, D., Chadwick, D., 2002. Verifiable credentials data model. [Online]: https://www.w3.org/TR/vc-data-model/. (Accessed May 22).
Sporny, M., Longley, D., Sabadello, M., Reed, D., Steele, O., C. Allen, 2023. Decentralized identifiers (DIDs) v1.0. [Online]: https://www.w3.org/TR/did-core/. (Accessed July 2023).
Thakkar, P., Natarajan, S., 2021. Scaling blockchains using pipelined execution and sparse peers. In: Proceedings of the ACM Symposium on Cloud Computing. Association for Computing Machinery.
Tian, L., Li, J., Li, W., Ramesh, B., Cai, Z., 2019. Optimal contract-based mechanisms for online data trading markets. IEEE Internet Things J. 6 (5), 7800–7810.
Zhang, X., Li, X., Miao, Y., Luo, X., Wang, Y., Ma, S., Weng, J., 2022a. A data trading scheme with efficient data usage control for industrial IoT. IEEE Trans. Ind. Inform. 18 (7).
Zhang, J., Yang, Y., Liu, X., Ma, J., 2022b. An efficient blockchain-based hierarchical data sharing for healthcare internet of things. IEEE Trans. Ind. Inform. 18 (10), 7139–7150.

Akanksha Dixit received her B.E. degree from the Army Institute of Technology, University of Pune, India. She is currently pursuing her M.Phil./Ph.D. degrees with the City, University of London, London, U.K. Her area of research includes digital identity and access management, Internet of Things security, blockchain, and applied cryptography.

Bruno Bogaz Zarpelão received his B.S. degree in Computer Science from State University of Londrina, Brazil, and the Ph.D. degree in Electrical Engineering from University of Campinas, Brazil. He is currently a professor at the Computer Science Department of the State University of Londrina (UEL), Brazil, which he joined in 2012. His research interests include security analytics, intrusion detection and Internet of Things.

Max Smith-Creasey received a B.Sc. (Hons) in Computer Science with Management from King' s College, University of London and a Ph.D. in Information Engineering from City, University of London. He has delivered technical solutions in domains ranging from telecommunications to insurance pricing. His research has revolved around enhancing authentication and identification technologies. He has published work in leading cyber security journals and international conferences. Several patent applications have been made based on his research.

Muttukrishnan Rajarajan received his BEng and Ph.D. degrees from City University of London in 1994 and 1999 respectively. He is the Founding Director of the Institute for Cyber Security at City, University of London and CEO of City defend Limited. Raj is a visiting researcher at British Telecommunications (BT) security research practice and continues to work closely with BT to supervise Ph.D. students. He was part of the team that worked on the UK Government' s Verify UK programme to contribute towards the privacy-by-design aspects of the overall architecture. Raj has extensive experience in working on cutting-edge commercial projects in the areas of encrypted search in the cloud, privacy-preserving data analytics and cloud security. He has published more than 350 publications, three books and hold two patents in cloud data privacy.