Annals of Telecommunications
https://doi.org/10.1007/s12243-026-01184-7

# An attribute-driven access control framework based on smart contracts for secure collaborative predictive maintenance

**Yago de R. dos Santos[1,2] · Ramon da Gama Cordeiro[1] · Yiannis Verginadis[3] · Diogo M. F. Mattos[2] · Marcela Tuler de Oliveira[1]**

Received: 20 March 2026 / Accepted: 4 May 2026


**Abstract**
Predictive maintenance systems rely on data sharing across organisations, yet commercially sensitive information requires precise access control to prevent competitive disadvantage. Existing centralised mechanisms require blind trust among participants, creating significant barriers to collaborative machine learning in industrial settings. This paper extends SDDK-AC (Secure Decentralised Data and Knowledge Access Control for Predictive Machinery Maintenance), an access control mechanism that couples Attribute-Based Access Control policies with blockchain and smart contracts, by implementing contextual attributes for geolocation verification and data integrity via hash comparison. The mechanism runs on a Hyperledger Besu permissioned blockchain, integrated with Keycloak and an Access Control Proxy. This paper evaluates 30,000 policy decisions across 30 experimental rounds, each comprising 1000 transactions, using a custom-developed Python evaluation script. The results show that most SDDK-AC functions achieve throughput above 60 transactions per second with an average latency of 14 ms, incurring approximately 16% overhead relative to a centralised ABAC baseline while still meeting predictive maintenance performance requirements.

**Keywords** Decentralized access control · Attribute-Based Access Control (ABAC) · Blockchain · Smart contracts · Predictive maintenance

## 1 Introduction

Data sovereignty has been defined as the right to maintain control over proprietary data and the intellectual property they imply [1]. Data sovereignty has become an incumbent strategy for application domains that rely on data-driven decision-making systems. Predictive maintenance (PdM) systems can significantly reduce unplanned downtime and repair costs when they learn from continuous streams of sensor readings collected from industrial equipment [2, 3]. In practice, these readings are often generated by distributed wireless sensors that must seamlessly cooperate to capture complementary aspects of machine health, including vibration, temperature, and acoustic emissions. Therefore, the effectiveness of PdM is closely related to the quality and completeness of sensor data collected across diverse operational environments and organisations.

However, the PdM models' reliable accuracy requires manufacturers and Machine Owners to frequently pool operational information. This creates a fundamental conflict of interest in competitive business environments. Organisations

Ramon da Gama Cordeiro, Yiannis Verginadis, and Diogo M. F. Mattos contributed equally to this work.

✉ Yago de R. dos Santos
yagorezende@id.uff.br

Ramon da Gama Cordeiro
R.ramonCordeiro@tudelft.nl

Yiannis Verginadis
jverg@aueb.gr

Diogo M. F. Mattos
menezes@midiacom.uff.br

Marcela Tuler de Oliveira
m.tulerdeoliveira@tudelft.nl

[1] Department of Engineering Systems and Services, Delft University of Technology (TU Delft), Delft, The Netherlands

[2] Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações (PPGEET), Universidade Federal Fluminense (UFF), Niterói, RJ, Brazil

[3] Athens University of Economics and Business/Institute of Communications and Computer Systems, Athens, Greece

Published online: 16 May 2026

operating in the same sector are usually competitors, making each participant in PdM scenarios reluctant to share any sensor data that could jeopardise its competitive advantage or even worse to leak confidential operational information. For example, sensor data can reveal details such as cycle times, cutting parameters, materials being processed, and tooling configurations, which competitors could take advantage of by reverse-engineering any such telemetry data towards discovering machines optimisation strategies, estimating production costs, or even replicating entire processes. Of course, if an organisation decides to avoid sharing such sensor data, then the outcome would lead to a stalemate, as valuable data remain siloed, model performance stagnates, while the economic benefits from PdM will remain unmet. Hence, a collaboration framework in a PdM is highly desired that grants each participating entity a verifiable and fine-grained control over *who* can access each resource, *for what purpose*, and *under what conditions*, without requiring blind trust in any single organisation.

The EU Data Governance Act (DGA) [4], implemented in September 2024, and the Data Act [5], which is active from September 12, 2025, have both established a legal framework for data sharing that reinforces the importance of such a collaboration model. These laws grant rights over data produced by sensor-enabled devices, favouring Machine Owners and imposing strict transparency, neutrality, and security requirements on Certified Data Intermediaries. Centralised solutions for data sharing and access control exist but require trust in third-party providers, often without verifiable transparency or compliance guarantees [6]. Decentralised solutions based on blockchain offer better alternatives, allowing machine learning stakeholders to share data transparently with fine-grained policies [7], yet important challenges remain, such as enforcing compliance, preserving resource integrity and data owner sovereignty, and supporting geolocation-based access control [8], while integrating with conventional web systems [9].

This paper extends our previous SDDK-AC (Secure Decentralised Data- and Knowledge Access Control for Predictive Machinery Maintenance) proposal [10], an access control mechanism that addresses these challenges by integrating Attribute-Based Access Control (ABAC) policies with blockchain and smart contract technologies. SDDK-AC continues the SmartAccess line of research [11] by recording ABAC policies and access decisions in a tamper-evident ledger, extending the original architecture to control access to heterogeneous sensor data used in industrial PdM machine learning. The proposed protocol operates in the context of dataset sharing (access control during the data capture phase is outside the scope), implements new policy rules for resource integrity verification through cryptographic hashes, and enforces geolocation-based restrictions through contextual attributes. The mechanism is implemented in a permissioned Hyperledger Besu network, integrated with Keycloak and an Access Control Proxy. The key contribution of the paper is to move from a proof-of-concept prototype [10] to a fully integrated and experimentally evaluated framework that incorporates new contextual policies for geolocation and hash-based integrity verification, together with an end-to-end deployment leveraging Hyperledger Besu, Keycloak, and an Access Control Proxy.

SDDK-AC provides three significant advances over the current state-of-the-art. First, unlike any other centralised access control mechanisms that usually rely on third parties [6], SDDK-AC registers all policies and access control decisions on a distributed ledger, ensuring transparency and auditability without any single points of trust. Second, while previous blockchain-based mechanisms such as SmartAccess [11] and the ABAC/XACML-based solution by Maesa et al. [7] do not implement geolocation or hash verification as contextual attributes, SDDK-AC explicitly incorporates both, enabling detection of unauthorised access attempts from wrong locations and identification of tampered resources. Third, in contrast to proposals that remain largely conceptual or lack integration with conventional systems [9, 12], SDDK-AC demonstrates practical feasibility through full implementation, including authentication, access control proxying, and performance benchmarking. Experimental results show that most SDDK-AC functions achieve throughput above 60 TPS, with an average latency of 14 ms, introducing approximately 18% overhead compared to a centralised Keycloak ABAC baseline while meeting predictive maintenance performance targets.

The remainder of this paper is organised as follows. Section 2 presents background concepts on IoT-based predictive maintenance, ABAC, and blockchain smart contracts. Section 3 reviews related work on decentralised access control mechanisms using blockchain. Section 4 describes the predictive maintenance scenario that motivates this research. Section 5 details the SDDK-AC mechanism, including architecture, authentication and authorisation flows, smart contract design, and security analysis. Section 6 presents the deployment environment and evaluation methodology, and Sect. 7 discusses the experimental results. Finally, Sect. 8 concludes the paper and outlines directions for future work.

## 2 Background

### 2.1 IoT and predictive maintenance

The Internet of Things (IoT) is a technology that integrates heterogeneous devices into a network, enabling them to communicate and to capture, store, and share data from sensors and smart devices [13]. Data is collected from mobile devices, such as smartphones, radio-frequency identifiers,

and wearables equipped with sensors [14]. The data are sent to a data storage facility and processed to interpret them.

Predictive maintenance is an evolution phase that follows reactive maintenance and precedes preventive maintenance. Those first two phases are not based on data analysis, and the development of predictive maintenance is made possible through the integration of industrial wireless sensors that collect data, such as temperature and vibration, with Machine Learning techniques to enhance operational efficiency, detect anomalies, and forecast maintenance [14].

### 2.2 ABAC

An access control policy is a set of rules that determines whether someone should be allowed or denied access to a resource. Typically, an access control authority defines and enforces the rules governing resource access [15]. There are many ways to implement access control, but in some cases, restrictions should be applied using multiple roles, particularly when sensitive data is involved. Thus, it is necessary to use attributes of the requesting person, those of the target resource, and, in some cases, contextual attributes.

Role-Based Access Control (RBAC) is an access control mechanism that permits or denies access based on a user's assigned roles. A group receives a role, and its users inherit the permissions associated with that role. All associated users lose access if a role's permissions change to revoke [16]. Although Attribute-Based Access Control (ABAC) is a dynamic, extensible, and fine-grained approach to implement access control, granting or denying access based on attributes related to role, resources, and context is not possible [11]. In contrast, ABAC is composed of five components of the XACML standard: Policy Administration Point (PAP), Policy Information Point (PIP), Policy Decision Point (PDP), Context Handler (CH), and Policy Enforcement Point (PEP) [17]. The first of these is PAP, which manages the policies used to evaluate access to resources. PIP manages context attributes from different resources. PDP evaluates the access request, gathering information from PAP, PIP, and CH to decide whether a user can access a specific resource. CH is responsible for translating and delivering processed data to PIP as context attribute values, particularly when those attributes originate from off-chain sources. PEP is responsible for enforcing or applying the result of an evaluation of the access request to the endpoint.

### 2.3 Blockchain and smart contracts

Blockchain is a decentralised data storage system, organised in a chain of blocks, which maintains data integrity and enables concise cryptographic operations, such as hashing, elliptic curves, and digital signatures [18]. In the blockchain, the data and the operations are available to network peers in a distributed ledger, which at least network members can read; in other cases, anyone can read it [19]. Each block contains a set of data that is added and validated by network partners. Data security and availability are ensured by hash and cryptographic operations, making tampering unlikely [20].

Smart contracts are self-executing, decentralised scripts that run on the blockchain and execute commands and operations defined in their code. Smart contracts, like blockchain operations, store results on the distributed ledger to track and audit operations, thereby protecting the network from misbehaviour [21]. The smart contract can be used to define and develop policies to enforce access control on a resource. It ensures that every access control rule is transparent because the code is stored in the blockchain. Moreover, the smart contracts ensure that the permit or denial resulting from a resource request is traced in the blockchain.

## 3 Related work

Most works in this area employ blockchain and smart contracts to implement access control. Some proposals adopt established standards such as ABAC [12] or RBAC [22], while others do not explicitly define the underlying access control model [23]. Standards are valued in academia and industry because they provide reliable and well-understood mechanisms for expressing and enforcing policies. These blockchain-based proposals typically store policies and execute access decisions via smart contracts, but they do not implement geolocation verification or hash-based integrity checking as contextual attributes, which limits their ability to enforce location-aware and integrity-aware access control.

De Oliveira et al. [11] propose SmartAccess, a decentralised access control mechanism that combines ABAC and smart contracts to manage cross-organisation medical record sharing. The proposal defines policies for two scenarios, patient appointments and acute situations, allowing requesters to access data within defined time frames, with access revocation triggered either by timestamp expiration or by the data owner. SmartAccess is directly related to SDDK-AC, as the latter extends its contract architecture to the predictive maintenance domain. The main difference is that SmartAccess does not implement geolocation and hash verification rules, whereas SDDK-AC introduces these contextual attributes to refine access control over sensitive sensor data that may reveal competitive business information.

Following the same research, Cordeiro et al. [10] introduce the initial proposal of SDDK-AC. In their proposal, access control policies and decisions are recorded on a tamper-evident ledger, and contextual attributes such as geolocation and hash-based data integrity are used to refine authorisation in a proof-of-concept implementation. The present paper builds directly on this prior work by generalising the original

architecture into a fully integrated framework with Keycloak-based authentication, an Access Control Proxy, deployment on a permissioned Hyperledger Besu network, and a comprehensive experimental evaluation to characterise throughput and latency under realistic workloads. However, Cordeiro et al. [10] presented only an initial proof-of-concept of SDDDK-AC without integration of more complex and realistic policy scenarios, whereas the present paper advances this line of research by demonstrating such integration within a fully implemented and experimentally evaluated framework.

Li et al. [23] design a domain-based mechanism for IoT data sharing, in which domain managers evaluate and manage access policies for their respective domains. This work addresses similar challenges to those considered in SDDK-AC, namely, secure sharing of sensor data in distributed environments. However, the access control model is not based on a standard framework such as ABAC or RBAC, which makes it harder to compare policy expressiveness and to integrate with existing identity and policy management tools. In contrast, SDDK-AC follows the ABAC model with clearly defined Policy Administration Point (PAP), Policy Decision Point (PDP), Policy Information Point (PIP), Context Handler (CH), and Policy Enforcement Point (PEP) components [17], facilitating interoperability and verification.

Tcydenova et al. [12] developed an access-control architecture for the IoT that uses oracles to transmit external data from the blockchain. Their work demonstrates how off-chain information can be incorporated into on-chain access control decisions, which is conceptually related to SDDK-AC's use of contextual attributes. However, their work provides limited details on the access control strategy and the security guarantees provided by the oracle mechanism, making it difficult to assess its robustness. In contrast, SDDK-AC explicitly defines contextual rules regarding geolocation and resource integrity, while it integrates them into a well-defined ABAC policy evaluation flow. This also incorporates an explicit threat model and security analysis.

Das and Namasudra et al. [24] propose an access control mechanism for healthcare data that combines Attribute-Based Encryption (ABE) with cloud storage. In their system, IoT devices capture health data, then they encrypt and store it using a cloud service provider, while data owners define access policies. Their work focuses on confidentiality and fine-grained cryptographic enforcement access control, but it remains centralised, therefore facing challenges such as efficient revocation control, limited transparency, and auditability. SDDK-AC differentiates from such research works by using blockchain technology to decentralise the policy persistence and decision logging, while providing transparent audit trails that rely on ABAC instead of ABE, regarding the policy evaluation.

Luding and Jiahao [25] present a Capability-Based Access Control (CP-BAC) mechanism that targets the edge computing setting using smart contracts. Their approach associates capabilities with users and resources, while it leverages blockchain to manage capability tokens at the edge. Although the work shows how blockchain can support distributed capability management, it does not adopt any ABAC-related capabilities or at least incorporate contextual attributes such as geolocation and integrity verification. SDDK-AC instead uses ABAC and context-based rules, which allow the definition of expressive policies tailored to predictive maintenance requirements.

Ohwo et al. [9] propose an access control mechanism designed for smart homes, combining ABE with differential privacy over the blockchain. Their proposal focuses on preserving privacy and introduces role abstractions similar to RBAC for managing home devices and services. However, the access control policies are not as fine-grained as those achievable with ABAC, and their integration with conventional web applications is quite limited. On the other side, SDDK-AC targets industrial environments, integrates with Keycloak and web APIs, and implements fine-grained ABAC rules, including geolocation and hash-based integrity checks.

Cruz et al. [22] present a cross-organisation RBAC mechanism deployed using smart contracts that allow a service provider to control access to resources over a public blockchain. A challenge–response protocol verifies whether a user holds the required role before granting access, while users can endorse other users to be assigned with roles. Their work demonstrates how RBAC roles can be propagated across organisations using blockchain, but the endorsement feature may be unsuitable for environments involving sensitive or strategic data. SDDK-AC follows a different approach by assigning policy control to consortium members (Machine Owners) and enforcing them based on attributes, and not just roles, which is more appropriate for governing access to sensitive operational data.

Ding et al. [26] propose a decentralised access control mechanism for the Internet of Things that simplifies ABAC through cryptographic proofs, enabling tamper detection in access requests. Their approach highlights the importance of combining cryptographic techniques with attribute-based decisions but introduces dependencies on Attribute Authorities that distribute private keys to IoT devices. SDDK-AC shares the goal of fine-grained attribute-based control but avoids such key distribution assumptions by relying on blockchain accounts and JWT-based authentication managed by Keycloak.

Maesa et al. [7] present a decentralised access control mechanism to manage access to smart contracts and digital assets using ABAC and XACML standards. The paper introduces an off-chain policy translator that converts XACML

policies into smart contract code and enforces them on-chain. This solution is closely related to SDDK-AC in its use of ABAC and smart contracts, but it keeps the PAP partially centralised, as policy storage and translation occur off-chain. SDDK-AC instead records policy associations and evaluation results directly on-chain and focuses specifically on predictive maintenance datasets, with contextual rules for location and integrity.

In follow-up work, Maesa et al. [8] enhance their previous proposal by integrating zero-knowledge proofs and Self-Sovereign Identity (SSI), enabling subjects to prove attribute possession without revealing the attributes. The approach removes intermediaries that request access on behalf of users and strengthens privacy guarantees, but it depends on trusted setup procedures in zero-knowledge proof systems, which can themselves become centralised points of failure. SDDK-AC is complementary to such identity-centric proposals, as it focuses on enforcing contextual access control over datasets in a consortium blockchain, rather than on designing privacy-preserving identity schemes.

Kim et al. [27] integrate ABAC with Decentralised Identity (DID) for access control in an energy transaction platform. Their work addresses privacy concerns in peer-to-peer energy markets by giving subjects control over their personal data and linking access decisions to decentralised identities. While this is related to SDDK-AC in its use of ABAC and decentralised identity concepts, the interaction between policy enforcement and DID is described at a high level, and the work does not cover geolocation or hash-based integrity checks. SDDK-AC instead concentrates on industrial predictive maintenance and provides a fully implemented mechanism with explicit contextual rules and performance evaluation.

Some proposals grant considerable power to end users, enabling them to store their own data [8] or to endorse other users [22]. In contrast, there are scenarios in which consortium members must strictly manage and monitor access control, particularly when data reveals strategic business information. SDDK-AC addresses such scenarios by building upon SmartAccess [11] to provide a fine-grained access control mechanism for predictive maintenance environments that receive data from sensors and smart devices. The proposal integrates blockchain, Keycloak, and a front-end application to deliver reliable access control for data sharing among peers in a competitive business context, and implements contextual attributes such as geolocation and hash-based integrity verification to enhance access control over predictive maintenance sensor data.

Overall, these approaches illustrate different trade-offs between decentralisation, privacy, and deployability. Compared with privacy-preserving identity approaches that rely on techniques such as zero-knowledge proofs or self-sovereign identity, SDDK-AC prioritises deployability and integration with conventional identity and access management infrastructures, such as Keycloak and OAuth-based authentication. While this design simplifies integration with existing industrial systems, it implies that certain identity attributes remain managed off-chain, potentially reducing the level of privacy achievable compared with fully decentralised identity models. Conversely, compared with traditional centralised ABAC systems, SDDK-AC introduces additional overhead from blockchain transaction processing but gains transparency, auditability, and resistance to unilateral policy manipulation by a single authority. These trade-offs reflect a design choice aimed at balancing practical deployability with decentralised trust guarantees in industrial collaborative environments.

## 4 Predictive maintenance

Predictive maintenance models for industrial machinery rely on continuous analysis of data generated by industrial machines that manufacturers supply to their different clients. These machines, deployed across various factories, generate valuable operational data that is used to develop Machine Learning models capable of detecting patterns associated with wear, anomalies, and potential failures. The data is periodically sent to a third-party entity, termed in this paper as *Data Processor*, which is a research entity responsible for handling and aggregating data from multiple *Machine Owners*. The *Data Processor* develops and trains advanced predictive models and deploys the resulting models back to each contributing machine, enabling local monitoring and early detection of issues across all installations, as shown in Fig. 1. In addition, the *Data Processor* supervises the overall performance of these models and updates them when necessary to maintain high levels of accuracy and reliability.

A key difficulty in this collaborative setting lies in granting secure and appropriate access to the operational data collected from all *Machine Owners*. The aggregated data are organised into datasets that train predictive algorithms which continuously monitor machine health and anticipate potential failures. On the one hand, each party should only have visibility into its own data and model results and not those of other organisations, while only the *Data Processor* analyst team is authorised to access the full cross-company dataset for algorithm development. On the other hand, the predictive model developed is standard for all *Machine Owners* that contributed to the training data, and when predictive accuracy falls below a threshold, the model is retrained centrally at the *Data Processor* and redeployed on-premises across all factories. In such cases, a more fine-grained access control mechanism is crucial towards facilitating the collaborative

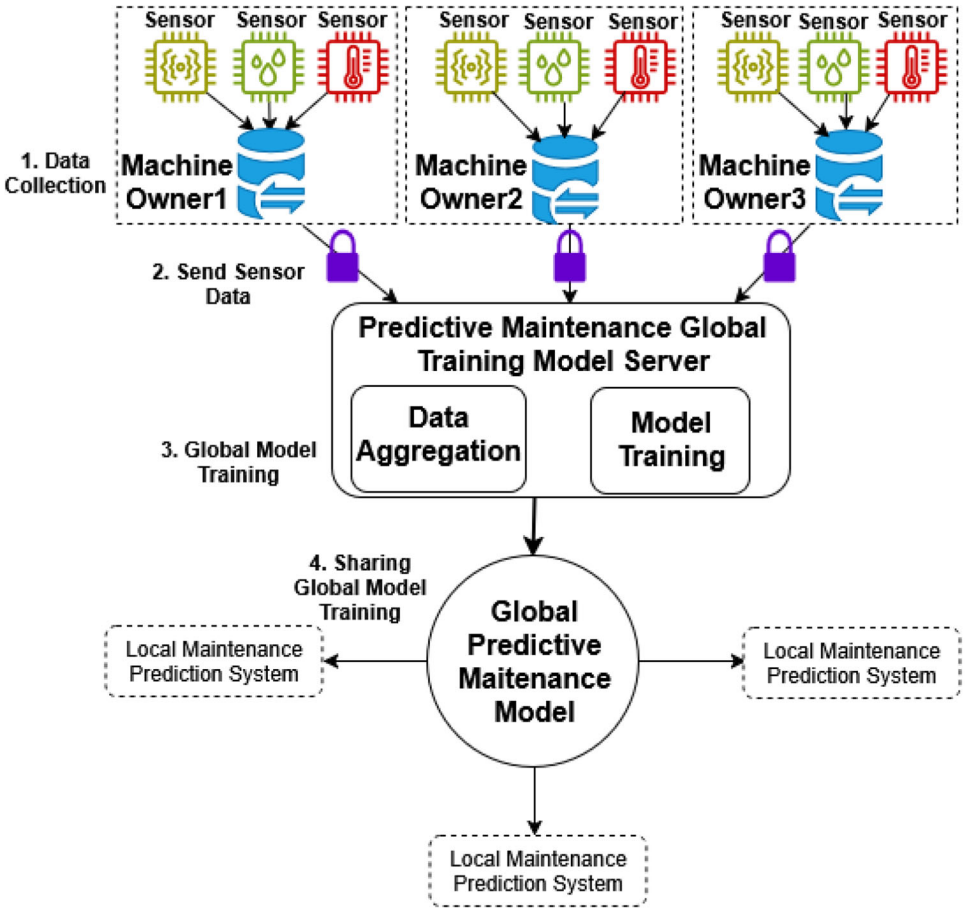



**Fig. 1** Predictive maintenance data lifecycle steps to generate a training model architecture. The coloured sensors, yellow, green, and red, represent different types of wireless sensors that collect data. Moreover, the Machine Owner's data are protected against unauthorised access

model development across different organisations, while preserving data sovereignty, privacy, and competitive confidentiality.

## 5 SDDK-AC mechanism

SDDK-AC extends SmartAccess [11] to address the access control requirements of collaborative predictive maintenance, using ABAC standards, blockchain, and smart contracts to implement and enforce policies for dataset sharing. The mechanism focuses on controlling access to heterogeneous sensor data and derived models, rather than on the data capture phase, which remains outside the scope of this work. New policy rules are introduced to provide more refined access control over sensitive data that may reveal business strategies. In particular, SDDK-AC implements resource integrity verification through cryptographic hashes and contextual geolocation restrictions, allowing policies to validate both the origin of the request and the integrity of the accessed resource. As a result, SDDK-AC fosters trustworthiness between the *Data Controller* and the *Data Processor* in a competitive industrial sector, enabling collaboration without requiring blind trust.

The hash-based integrity verification rule ensures that the accessed resource matches the expected representation registered in the system. During the authorisation process, the hash of the requested resource is compared with the reference hash stored in the access control policy context. If the values differ, the request is denied, preventing access to resources that may have been modified or tampered with. It is important to note, however, that hash verification guarantees only integrity with respect to the stored representation. It does not by itself ensure properties such as data freshness, provenance, or semantic correctness. For example, a dataset may remain unchanged even as it becomes outdated, or it may originate from an unreliable source. Addressing these aspects would require complementary mechanisms such as provenance tracking, trusted data pipelines, or timestamp-based validation. Therefore, in SDDK-AC, hash verification is intended primarily to detect unauthorised modification of protected resources.

The implementation of SDDK-AC includes both on-chain and off-chain components that together provide a complete and deployable mechanism. On-chain, smart contracts encode access control policies and evaluate access requests under the ABAC model. Off-chain, an Access Control Interface (ACI), an Access Control Proxy (AC Proxy), and

Keycloak provide user management, authentication, and integration with the predictive maintenance environment. This combination makes it easier to utilise blockchain technology in real scenarios and delivers a transparent and auditable access control mechanism that integrates with existing web-based infrastructures.

### 5.1 SDDK-AC architecture

The SDDK-AC architecture comprises three main components. The first two are off-chain components, namely the Access Control Interface (ACI) and the Access Control Proxy (AC Proxy), which integrate with the Execution Engine (EE) and the Data Management (DM) services to protect data processing. The third component is the institutional node, which contains a Keycloak instance and a blockchain node for each participating *Machine Owner*. Each *Data Controller* operates its own Keycloak, integrated with the off-chain components to provide JSON Web Tokens (JWTs) and identity management, while the blockchain hosts a set of smart contracts that enforce ABAC-based rules.

Figure 2 illustrates the overall architecture and the interactions among components. Steps 1 and 2 (in grey) correspond to the authentication mechanism based on the OAuth 2.0 standard, whereas the steps highlighted in light orange represent the authorisation mechanism. The ACI exposes a REST API, a User Management module, and a Blockchain Interface layer that bridges users from the Web 2 application to the Web 3 environment. The ACI supports user registration and authentication, retrieval of user attributes, and assignment of access-related attributes through the integration with the Frontend application and Keycloak. Upon registration,

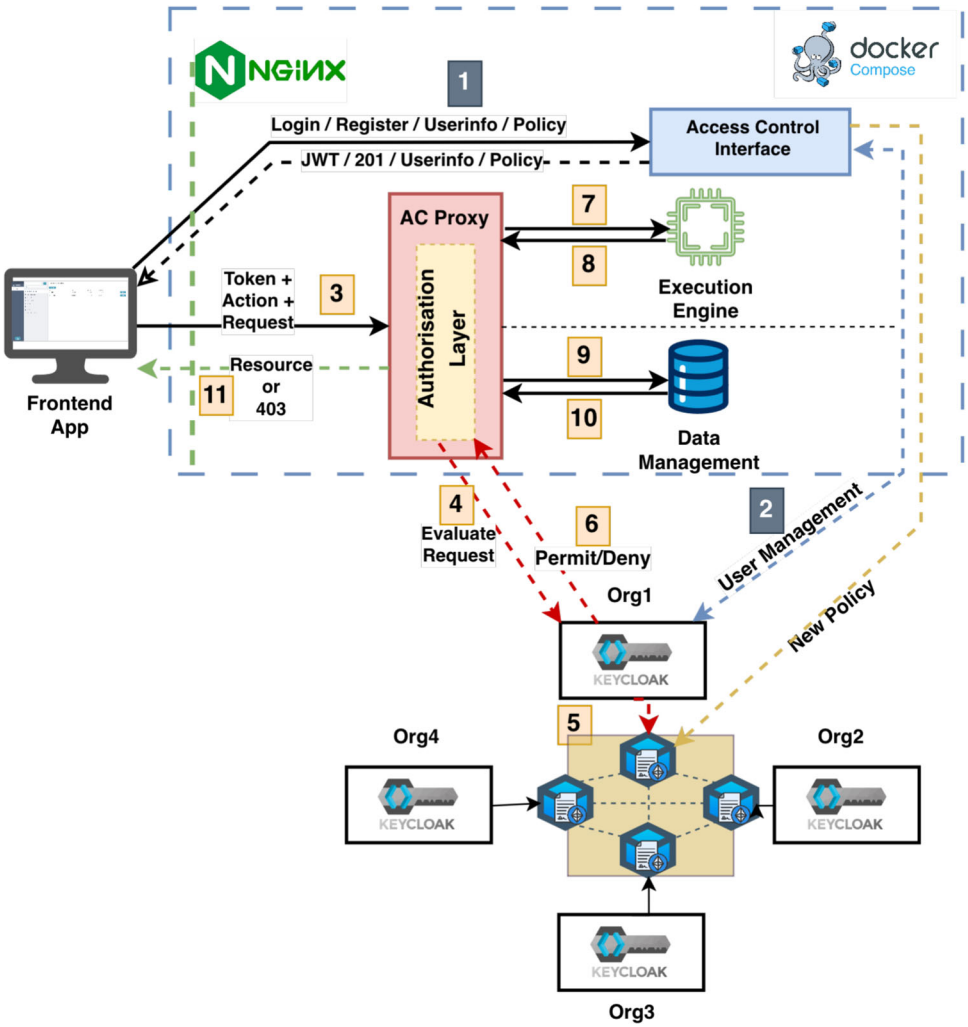



**Fig. 2** SDDK-AC mechanism architecture and data flow. The figure illustrates the interaction between the ACI, the protected services behind the AC Proxy, the organisations' Keycloak instances, and Hyperledger Besu nodes. Steps 1 and 2, in grey, are part of the authentication mechanism using the OAuth 2.0 standard. Steps 3 to 11, in light orange, correspond to the authorisation mechanism

user information is persisted in the organisation's Keycloak, which issues a JWT token for each authenticated user. It is important to note that our solution doesn't persist or store any credentials or other Personally Identifiable Information (PII) on the blockchain; instead, it maintains only a list of users' wallet addresses and access-control-related data, while all other user attributes remain off-chain in the institutional Keycloak database. This separation of concerns preserves privacy while it enables distributed access control decisions.

The security guarantees of the framework are provided by the interaction between its main components. Keycloak is responsible for authentication, identity management, and secure issuance of JWT tokens. It ensures that users are correctly identified before any authorisation request is evaluated. The blockchain infrastructure guarantees the integrity, transparency, and immutability of access control policies and decisions. Smart contracts deterministically evaluate requests and persist the results on a distributed ledger. The AC Proxy acts as the Policy Enforcement Point (PEP). It ensures that protected services cannot be accessed unless a valid authorisation decision is obtained from the blockchain. Together, these components establish clear trust boundaries: identity assurance is delegated to Keycloak, policy evaluation and auditability are ensured by the blockchain, and access decision enforcement is handled by the proxy layer.

To be able to exploit contextual attributes related to geolocation in access control decisions, SDDK-AC implements three dedicated context handlers. The first handler, the *IP Location Checker*, verifies whether the requester's IP address corresponds to an expected (by a policy) city, providing a valuable coarse-grained location validation. The second handler, the *Distance Checker*, verifies that two geographic coordinate pairs lie within a predefined radius, enabling fine-grained proximity checks. The third handler, the *IP-to-Coordinates Converter*, obtains approximations of latitude and longitude, coming from an IP address, by using an IP geolocation service. The provided location is used by the other handlers only when an IP address is available. These handlers are integrated with SDDK-AC through the Context Handler API, which allows smart contract policies to dynamically evaluate the requester's location at run-time and to enforce geolocation-based access control rules. Together, they provide the contextual data required to evaluate geolocation attributes in the ABAC paradigm.

Although IP-based geolocation provides a practical contextual signal for access control decisions, it is inherently approximate. Factors such as Network Address Translation (NAT), corporate proxies, VPN usage, and mobile routing may affect the accuracy of the inferred location. Consequently, the geolocation policies implemented in SDDK-AC should be interpreted as coarse-grained contextual constraints rather than precise physical location verification. In scenarios requiring stronger guarantees, additional mechanisms such as device attestation, secure GPS signals, or trusted location oracles could complement IP-based methods.

### 5.2 Authentication flow

The ACI manages user authentication, user registration, and policy deployment, and it also acts as a bridge between the Web 2 Frontend application and the Web 3 blockchain environment. The authentication flow starts when the user logs in through the Frontend (step 1 in Fig. 2). The Frontend forwards the login request to the ACI, which in turn authenticates the user against the organisation's Keycloak instance (step 2). Once a user is registered and their provided credentials are valid, Keycloak issues a JWT access token, which is then securely returned to the user through the Frontend application. All other user-related tasks, such as registration and attribute management, follow the same interaction pattern, differentiating only with respect to the specific input and output data.

This proposed design distinguishes authentication from authorisation tasks, while maintaining a clear interaction between the two. Keycloak is responsible for identity and attribute management, while the blockchain is responsible for policy storage and evaluation. Based on this, users authenticate using familiar web-based mechanisms and then use JWTs to obtain authorisation decisions mediated by the AC Proxy and enforced by smart contracts. This separation enhances security and makes the system easier to adopt in environments that already rely on standard identity and access management solutions.

### 5.3 Authorisation flow

The authorisation flow, depicted in Fig. 2, describes the way SDDK-AC grants or denies access to an incoming request concerning a protected resource. The process starts when a user with a valid JWT requests access to a resource through the Frontend, and the request is intercepted by the AC Proxy (step 3). The AC Proxy checks whether the requested *ResourceID*, which denotes the unique identifier of the resource, has an access control policy associated with it. If a policy exists, the AC Proxy forwards the user request and the resource's hash (used as the identifier in the Data Management) to Keycloak for evaluation (step 4).

In the next step, Keycloak constructs a transaction containing the relevant request information, including subject, resource, action, and contextual attributes, and submits it to the blockchain for policy evaluation (step 5). A sequence of smart contracts evaluates the policies and, once the evaluation is complete, emits an event with the decision result.

The smart contract broadcasts this event on-chain, and Keycloak listens for it and retrieves the decision when it becomes available. Keycloak then returns the decision, either `Permit` or `Deny`, to the AC Proxy (step 6), which acts as the Policy Enforcement Point (PEP).

If access is permitted, the AC Proxy provides the user UUID, the *ResourceID*, and a token with the necessary permissions to the Execution Engine (EE) so that it can access the Data Management service (DM) (step 7). The EE then requests the resource, and this request is routed through the AC Proxy, because the EE and DM do not communicate directly (step 8). The AC Proxy forwards the EE request to the DM (step 9), the DM returns the resource to the AC Proxy (step 10), and finally the AC Proxy delivers the resource to the user (step 11). If access is denied, the AC Proxy returns an error response, and the EE is not triggered to process the data.

SDDK-AC assumes that all data is securely stored in the DM and that data processing occurs only within the secure EE. Furthermore, processing is triggered only after the AC Proxy receives a positive evaluation result from the blockchain. The AC Proxy is location-agnostic because policies are stored and enforced on the blockchain, so the physical location or redundancy of AC Proxy instances does not affect correctness. This property improves system fault tolerance and allows multiple AC Proxy instances to connect to redundant Keycloak instances and blockchain nodes.

## 5.4 Blockchain network and smart contracts

The blockchain network nodes represent the *Machine Owners*, who are responsible for defining consensus parameters, specifying access control policies for their resources, auditing requests, and enforcing their policies. Policies are encoded as smart contracts and associated with the specific resources that require protection, using the resource's URI as the *ResourceID*. The blockchain network provides transparency and auditability in the access control flow, thereby fostering trust between entities that define access control policies (the *Machine Owners*) and entities that request access (the *Data Processors*). In data-sharing environments, when competitors are involved, this is particularly important since predictive maintenance sensor data can reveal strategic business-related information. Instead, there should be strong guarantees implemented regarding data sovereignty and access control.

In this context, blockchain offers properties that are well aligned with SDDK-AC's goals. The permissioned network ensures that only the authorised institutions operate nodes, while the smart contracts provide deterministic and verifiable execution of access control logic. Hence, every access request evaluation is persisted as a new transaction on the distributed ledger, which makes policy evaluation traceable and auditable over time. This approach allows consortium members to verify that access control rules are enforced consistently across the network and that there is no single party that can unilaterally alter the evaluation logic.

### 5.4.1 Smart contracts-based approach

The smart contracts in SDDK-AC involve policy point contracts and dedicated *RulesContracts*, following the same general structure introduced in SmartAccess [11]. These policy point contracts consist of the `Policy Decision Point Smart Contract (PDPSC)`, `Policy Administration Point Smart Contract (PAPSC)`, and `Policy Information Point Smart Contract (PIPSC)`. In addition, there is a dedicated *RulesContract* that encodes the access control rules for the predictive maintenance use case described in Sect. 4. Figure 3 illustrates the interactions between these contracts.

These smart contracts interact so that they can yield a decision on whether to permit or deny access to an incoming request for a specific protected resource. The result of the decision is returned directly to the PEP, which in this architecture consists of a Keycloak instance and the AC Proxy. The resource contains a URI that serves both as a means of accessing the resource and as its unique identifier, i.e., *ResourceID*. This identifier must be protected, and the access to the corresponding resource must be granted only if a permit decision has been yielded by the evaluation of policies associated with its *RulesContract*. The `PAPSC` maintains a mapping of protected *ResourceIDs* to the addresses of their associated *RulesContracts*, which allows the `PDPSC` to locate the appropriate contract to be enforced for each access request.

When the PEP forwards an incoming access request to the blockchain, the `PDPSC` receives the request and queries the `PAPSC` to obtain the address of the *RulesContract* that protects the target *ResourceID*. The `PDPSC` then invokes the corresponding *RulesContract* to evaluate the incoming request attributes against the policy rules, which may include subject attributes (e.g., organisation or role), resource attributes (e.g., dataset type), action attributes (e.g., read or write), and contextual attributes (e.g., geolocation or hash integrity). During this process, the `PIPSC` provides additional attribute values when required, including contextual data obtained through the available context handlers. After all relevant rules have been evaluated, the *RulesContract* returns the decision (`Permit` or `Deny`) to the `PDPSC`, which emits an event with the final result. Keycloak subscribes to such events, retrieves the decision result, and passes it to the AC Proxy to complete the authorisation flow.

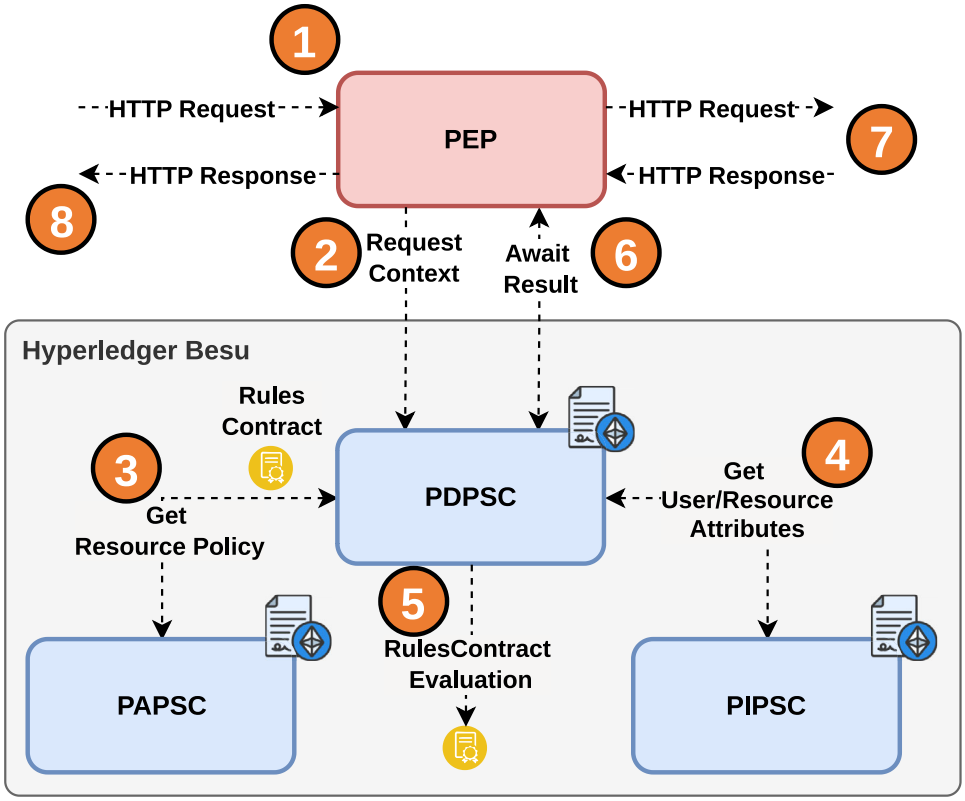



**Fig. 3** The access control request flows between the off-chain and on-chain components, starting from the `Policy Enforcement Point (PEP)` to the evaluation result. In the diagram, the off-chain module is indicated by the red box, while the on-chain modules, which are developed as smart contracts, are represented by the blue boxes. The `RulesContract` is denoted by the yellow icon

This smart contract design separates the logic of policy decisions from the actual policy administration and information provisioning while mirroring the ABAC paradigm and facilitating updates and extensions. New *RulesContracts* can be deployed for additional resources or scenarios without modifying existing contracts, and the `PAPSC` can be updated to map new *ResourceIDs* to their corresponding rule sets. Such modularity allows end-users to evolve their access control policies while preserving interoperability and auditability within the SDDK-AC framework.

## 6 SDDK-AC deployment and evaluation

The SDDK-AC prototype implementation and deployment were executed on a server with an Intel Xeon Platinum 8358P CPU, operating at 2.60 GHz, and 16 GB of RAM. The system runs Ubuntu 22.04 LTS (Jammy Jellyfish) as the operating system, with a 160 GB solid-state disk. The software environment includes Docker and Docker Compose for containerisation and NGINX version 1.26 to protect Keycloak and facilitate safe request redirection. Keycloak version 21.1.2 was used for authentication and authorisation management. Furthermore, Python 3 is used for scripting and automation purposes. The experimental policy enforcement scenario design defined the following stages: (i) rule definition; (ii) identification of contextual attributes; (iii) smart contract implementation; and (iv) integration with Keycloak and the Context Handler API. The experiment with 30 rounds and 1000 transactions per round was conceived to exercise precisely this complete policy, thereby validating that the mechanism correctly enforces all specified rules under realistic workload conditions. For blockchain operations, the Hyperledger Besu version 24.01 framework is utilised to launch a consortium blockchain with four nodes, running on OpenJDK 19. The smart contracts were written in the Solidity language version 0.8.20.

The experimental setup was run on a single server that hosted all blockchain nodes and system components. This configuration allows controlled comparison with the centralised baseline and isolates the computational overhead introduced by the decentralised authorisation mechanism. In real consortium deployments, blockchain nodes are typically distributed across different organisations and geographic locations, which introduces additional network latency during transaction propagation and consensus. In such scenarios, the end-to-end authorisation latency would include both the smart contract execution time and the network delay associated with inter-node communication. Although the present evaluation does not simulate wide-area network conditions, permissioned blockchain platforms such as Hyperledger Besu are designed to operate efficiently in geographically distributed environments.

Access Control Interface and AC Proxy are deployed using GitHub Actions to create public Docker images. These Docker images can be used to set up the architecture outlined in Fig. 2. Additionally, the figure illustrates that all services are located behind an Nginx proxy, with Docker Compose serving as the orchestrator for all services. Docker Compose is a tool for defining and running multi-container Docker applications, allowing for easy deployment and scaling of the system. All the module's details are presented in further sections. The flow illustrated in Fig. 2 outlines a sequential series of actions for authentication and authorisation.

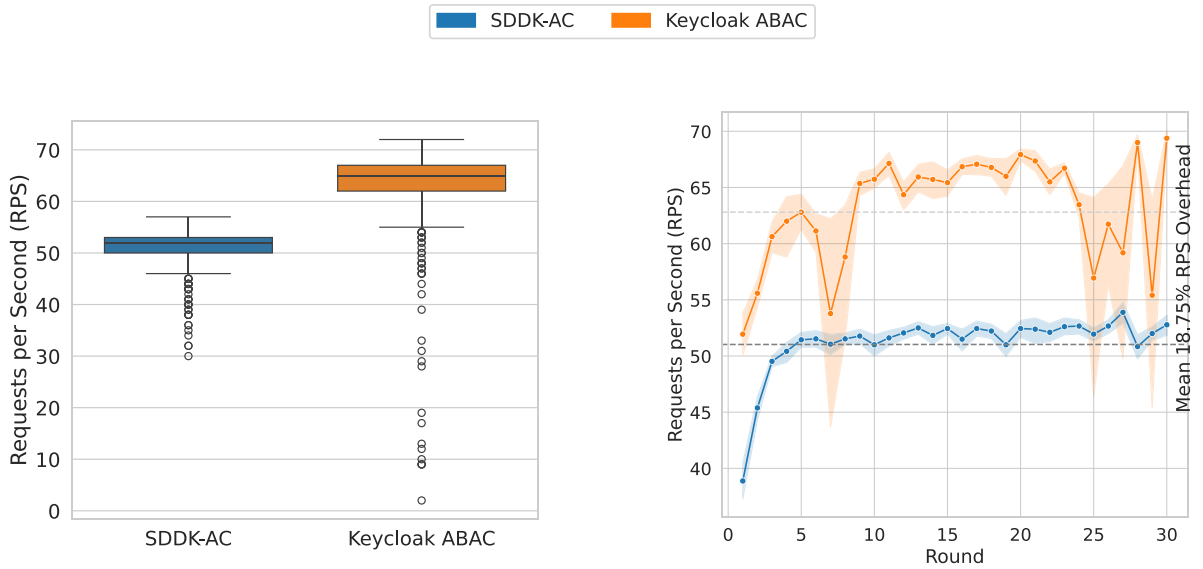
Figure 3 shows the flow of the request. The smart contracts were implemented using the Solidity language. The implementation was designed to follow a flow of actions; when the request arrives, it reaches `PEP` (step 1), and from `PEP`, it first reaches the policy point contracts. The `PEP` communicates only with the `PDPSC` to protect the other smart contracts.

Thus, `PDPSC` receives the request and its context attributes through the `evaluateRequest` function (step 2), which is an interface to call `RulesContract` later. `PDPSC` manages this request by calling `PAPSC` with the URI of the resource. After that, `PDPSC` waits for the `PAPSC` to send the `RulesContract` address (step 3), which has all the functions to evaluate the request. Additionally, `PDPSC` provides as parameters the resource URI and the user's address to the `PIPSC`, which returns the non-identifiable context attributes of the resource and the user (step 4). Once `PDPSC` has the resource's context attributes and `RulesContract` address, it calls the *RulesContract*, providing the resource's context attributes and the user's attributes to `RulesContract` to evaluate the access using the function `evaluateRequest` implementation, evaluating rule by rule. After the evaluation, the `RulesContract` sends the result to the `PDPSC` using the `evaluateRequest`(step 5). Finally, with the result, `PDPSC` emits it, and `PEP` can catch that result.

## 7 Results and discussion

### 7.1 Comparison with centralised access control baseline

We evaluated the temporal overhead of the complete SDDK-AC policy implementation compared to a centralised ABAC mechanism in Keycloak, considered as a Web 2 baseline. For this assessment, the predictive maintenance use case policy was encoded both in a Solidity smart contract and as a JavaScript policy in Keycloak, and executed over 30 rounds with 1000 requests per round. Figure 4a presents the distribution of Requests per Second (RPS) achieved by both mechanisms, revealing that the decentralised SDDK-AC consistently sustains lower RPS values than the centralised Keycloak ABAC baseline, reflecting the additional cost of executing access control decisions as blockchain transactions. Nonetheless, the obtained RPS values remain within the range required by predictive maintenance use cases, indicating that the decentralised approach is compatible with realistic workloads. Figure 4b confirms that, across the 30 rounds, SDDK-AC exhibits more stable behaviour with lower relative RPS variation, whereas Keycloak ABAC




(a) RPS distribution over 30 rounds.

(b) RPS per round with 95% confidence intervals.

**Fig. 4** Requests per Second (RPS) of SDDK-AC and Keycloak ABAC. **a** shows the RPS distribution across 30 rounds with 1000 requests per round, whereas **b** depicts the evolution of RPS per round, including mean values and 95% confidence intervals

presents more pronounced fluctuations between rounds, which prepares the discussion on the impact of this variability on transaction latency.

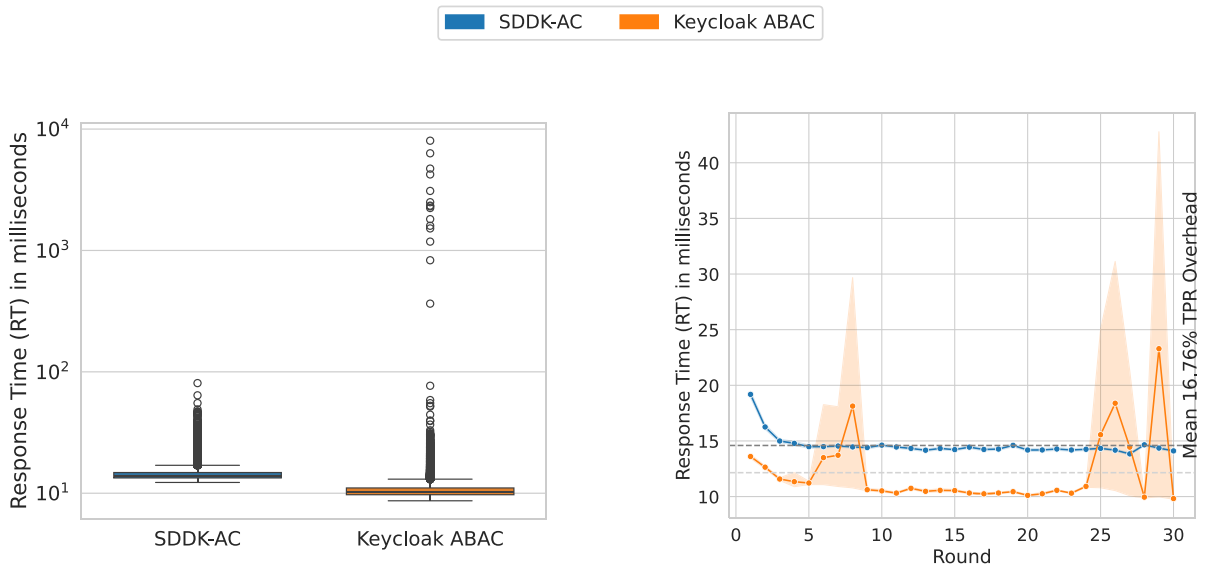
The evaluation results indicate that SDDK-AC exhibits an average Response Time (RT) of approximately 14 ms, while Keycloak ABAC achieves approximately 12 ms, resulting in an average overhead of about 16.76% in RT. In addition, it presents an average overhead of approximately 18.75% in RPS terms. This difference reflects the additional cost of distributed consensus, transaction ordering, and validation inherent to blockchain-based access control, while simultaneously confirming that the decentralised mechanism remains viable for predictive maintenance scenarios with low-latency requirements. Figure 5a presents the distribution of RT on a logarithmic scale, showing that the median RT of SDDK-AC is slightly higher than that of Keycloak ABAC, yet both remain in the order of tens of milliseconds. Figure 5b demonstrates that SDDK-AC exhibits more predictable RT variation across the 30 rounds, whereas Keycloak ABAC alternates between rounds with significantly lower and higher latencies, suggesting that distributed access control introduces a fixed additional cost that is, however, more stable in delivering the access decision.

## 7.2 Latency breakdown of policy evaluation stages

To better understand the sources of latency in the SDDK-AC authorisation pipeline, we performed an additional latency breakdown analysis. The measured latency corresponds to the policy evaluation workflow illustrated in Fig. 3, which presents the XACML-based ABAC model adapted for blockchain environments. In this analysis, the total response time is decomposed into two main stages. The Context Handler stage corresponds to steps 1 and 2 of Fig. 3, where the request is intercepted, and the contextual attributes required for policy evaluation are assembled. This stage includes retrieving user attributes and contextual information needed to build the evaluation context. The Evaluate Request stage corresponds to steps 3–6 in Fig. 3, which involve the policy evaluation process executed on the blockchain. In particular, this stage measures the execution latency of the `evaluateRequest()` function described in Sect. 6, which evaluates the access control rules implemented as smart contract functions and produces the final authorisation decision.

In the experiments, geolocation data was already registered in the user attribute repository and retrieved directly during contextual attribute resolution. Therefore, the reported results do not represent IP-based geolocation inference. In practical deployments, geolocation derived from IP addresses may require external services or APIs to infer geographic coordinates, which can introduce additional latency. Analysing the performance impact of such external geolocation services is outside the scope of this work and remains an important direction for future evaluation.

Figure 6 presents the mean latency contribution of each stage across the experimental rounds, with a 95% confidence interval. The results shown in Fig. 6 indicate that the Evaluate Request stage accounts for the largest portion of the total response time, representing 65.4% of the latency,




(a) RT distribution over 30 rounds.

(b) RT per round with 95% confidence intervals.

**Fig. 5** Response Time (RT) of SDDK-AC and Keycloak ABAC. **a** highlights the central tendency and outliers in RT, while **b** shows the evolution of RT per round, evidencing the more predictable behaviour of the decentralised mechanism

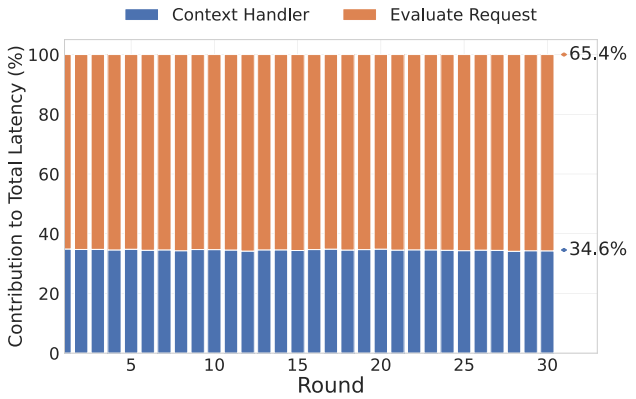



**Fig. 6** Latency breakdown of the SDDK-AC authorisation pipeline. The chart shows the percentage contribution of the two main stages of the policy evaluation process: Context Handler, responsible for contextual attribute resolution, and Evaluate Request, corresponding to smart contract policy evaluation. The reported values represent the mean latency across the experimental rounds, with a 95% confidence interval, illustrated by the circle shown in the right corner of the figure

while the Context Handler stage contributes the remaining 34.6%. This behaviour is expected because the policy evaluation phase includes executing smart contract logic and the associated blockchain transaction lifecycle, which involves transaction submission, validation, and event notification within the permissioned network. In contrast, the Context Handler stage mainly performs contextual attribute retrieval and preparation, which are comparatively lightweight operations when the required attributes are already available in the user attribute repository.

These results suggest that the majority of the latency introduced by SDDK-AC originates from the blockchain-based policy enforcement process rather than from contextual attribute resolution. This observation is consistent with the expected overhead of distributed ledger operations, where consensus and transaction processing introduce additional delay compared to purely off-chain mechanisms.

## 8 Conclusion

This paper addressed the challenge of secure data sharing for collaborative predictive maintenance in competitive industrial environments, where Machine Owners must balance the benefits of collective machine learning against the risks of exposing commercially sensitive operational data. The proposed SDDK-AC (Secure Decentralised Data- and Knowledge Access Control for Predictive Machinery Maintenance) mechanism combines Attribute-Based Access Control policies with blockchain and smart contracts, implementing contextual attributes for geolocation verification and resource integrity checking through cryptographic hash comparison. By recording access control policies and decisions on a tamper-evident distributed ledger, SDDK-AC enables each *Data Controller* to share data, manage access, maintain data sovereignty, and enhance trust among partners, while integrating with off-chain components such as Keycloak and a Frontend application to provide practical usability. A comparative assessment against a centralised Keycloak ABAC baseline further indicated that SDDK-AC incurs an overhead of approximately 10–18% in time per transaction and about 18.75% in RPS terms while remaining compatible with predictive maintenance performance requirements.

Future work will focus on several directions to further improve SDDK-AC. First, we plan to optimise the performance of geolocation functions and modify the `evaluateRequest` logic to short-circuit to `false` upon the first attribute mismatch, enabling comparison with the current constant-time design. Second, we intend to deploy SDDK-AC on other blockchain platforms, such as Hyperledger Fabric, to analyse platform-specific performance characteristics. Third, we will investigate the use of Oracles to manage contextual attributes, with improved security and performance, and compare on-chain and Oracle-based approaches. Finally, future work will analyse the impact of cross-organisational deployments on system performance, including the effects of network delay, increasing consortium size, and the computational overhead introduced by contextual policy rules such as geolocation and hash verification.

Another promising research direction involves harnessing artificial intelligence (AI) to advance blockchain-based infrastructures for access control and trust management. Recent studies have demonstrated that machine learning mechanisms can interact directly with blockchain protocols to enhance system behaviour in a variety of ways, including supporting adaptive trust models, detecting anomalies in distributed environments, and optimising policy decisions [28]. Within the realm of collaborative predictive maintenance, these techniques could enable dynamic access control policies that respond intelligently to system behaviour, proactively identify abnormal access patterns, and refine policy rules based on securely recorded historical decisions stored on the blockchain. This evolving perspective reflects a growing body of work on AI-enhanced blockchain systems, which shows that AI can improve the effectiveness and efficiency of blockchain-based applications by analysing system interactions, optimising protocol performance, and discovering novel trust-management strategies in decentralised settings. By integrating these innovative approaches with SDDK-AC, it may be possible to further strengthen the resilience, adaptability, and intelligence of decentralised access control frameworks.

**Author contributions** Y. R. dos Santos and R. da Gama Cordeiro contributed to software, implementation, validation, formal analysis, investigation, and data curation, being responsible for the implementation of the proposed system and the execution of the experimental evaluation. M. T. de Oliveira contributed to conceptualization, methodology, funding acquisition, and supervision; Y. Verginadis contributed to funding acquisition and supervision, both supporting the design of the research and securing the funding that enabled this work. D. M. F. Mattos contributed to writing—original draft, writing—review and editing, validation, and supervision, being responsible for the preparation of the manuscript and the critical review and validation of the results. All authors reviewed and approved the final version of the manuscript.

**Funding** This study was partially funded by the European Union's Horizon Program under grant agreement no. 101093164 (ExtremeXP: EXPerimentation driven and user eXPerience oriented analytics for eXtremely Precise outcomes and decisions).

**Data availability** All experimental results and scripts used in this study are available in an open-access repository at: https://github.com/yagorezende/SDDK-AC.

## Declarations

**Ethics approval** Not applicable.

**Consent to participate** Not applicable.

**Consent for publication** Not applicable.

**Competing interests** The authors declare no competing interests.



## References

1. Scherenberg F, Hellmeier M, Otto B (2024) Data sovereignty in information systems. Electron Mark 34(1):15
2. Hossain M, Sanim S, Kamruzzaman S, Yesmin M, Sayem MS, Shufian A (2025) Anomaly detection in industrial machinery using machine learning and deep learning techniques with vibration data for predictive maintenance. In: 2025 2nd International conference on advanced innovations in smart cities (ICAISC). IEEE, pp 1–6
3. Narayanan LK, Loganayagi S, Hemavathi R, Jayalakshmi D, Vimal V (2024) Machine learning-based predictive maintenance for industrial equipment optimization. In: 2024 International conference on trends in quantum computing and emerging business technologies. IEEE, pp 1–5
4. European Commission: European Data Governance Act. https://digital-strategy.ec.europa.eu/en/policies/data-governance-act. Accessed: 03 Jul 2025; entered into force on 23 June 2022, applicable since September 2023
5. Kiteworks: introduction to the EU Data Governance Act. https://www.kiteworks.com/risk-compliance-glossary/eu-data-governance-act/. Accessed: 03 Jul 2025
6. Wang Y, Ma Y, Xiang K, Liu Z, Li M (2018) A role-based access control system using attribute-based encryption. In: 2018 International conference on big data and artificial intelligence (BDAI). IEEE, pp 128–133
7. Maesa DDF, Mori P, Ricci L (2019) A blockchain based approach for the definition of auditable access control systems. Comput Secur 84:93–119
8. Maesa DDF, Lisi A, Mori P, Ricci L, Boschi G (2023) Self sovereign and blockchain based access control: supporting attributes privacy with zero knowledge. J Netw Comput Appl 212
9. Ohwo OB, Ajayi W, Udosen A, Amusa A, Agyei MY, Bamidele O (2024) Hybrid privacy-preserving access control mechanism using blockchain and attribute-based access control for smart home. In: 2024 IEEE SmartBlock4Africa. IEEE, pp 1–8
10. Gama Cordeiro R, Santos Y, Verginadis Y, Tornos B, Eizaguirre GG, Mattos DMF, Oliveira MT (2025) SDDK-AC: secure decentralised data-and-knowledge access control for predictive machinery maintenance. In: 2025 13th Wireless Days Conference (WD), pp 1–9. https://doi.org/10.1109/WD67713.2025.11302762
11. De Oliveira MT, Reis LHA, Verginadis Y, Mattos DMF, Olabarriaga SD (2022) SmartAccess: attribute-based access control system for medical records based on smart contracts. IEEE Access 10:117836–117854
12. Tcydenova E, Seok B, Cho M, Lee C (2022) Decentralized access control for internet of things using decentralized identifiers and multi-signature smart contracts. In: 2022 International conference on platform technology and service (PlatCon). IEEE, pp 66–70
13. Nguyen DC, Ding M, Pathirana PN, Seneviratne A, Li J, Niyato D, Dobre O, Poor HV (2021) 6G Internet of Things: a comprehensive survey. IEEE Internet Things J 9(1):359–383
14. Haque R, Bajwa A, Siddiqui NA, Ahmed I (2024) Predictive maintenance in industrial automation: a systematic review of IoT sensor technologies and AI algorithms. Am J Interdisc Stud 5(01):01–30
15. Oliveira MT, Verginadis Y, Reis LH, Psarra E, Patiniotakis I, Olabarriaga SD (2023) AC-ABAC: attribute-based access control for electronic medical records during acute care. Expert Syst Appl 213
16. Hlushchenko P, Dudykevych V (2024) Exploratory survey of access control paradigms and policy management engines. In: Proceedings of the seventh international workshop on computer modeling and intelligent systems (CMIS-2024). CEUR Workshop Proceedings, vol 3702, pp 192–202. https://ceur-ws.org/Vol-3702/paper22.pdf
17. Psarra E, Verginadis Y, Patiniotakis I, Apostolou D, Mentzas G (2021) Accessing electronic health records in critical incidents using context-aware attribute-based access control. Intell Dec Technol 15(4):667–679
18. Yue K, Zhang Y, Chen Y, Li Y, Zhao L, Rong C, Chen L (2021) A survey of decentralizing applications via blockchain: the 5G and beyond perspective. IEEE Commun Surv Tutor 23(4):2191–2217
19. Oliveira MT, Carrara GR, Fernandes NC, Albuquerque CV, Carrano RC, Medeiros DS, Mattos DM (2019) Towards a performance evaluation of private blockchain frameworks using a realistic workload. In: 2019 22nd Conference on innovation in clouds, internet and networks and workshops (ICIN). IEEE, pp 180–187
20. Oliveira MT, Reis LH, Medeiros DS, Carrano RC, Olabarriaga SD, Mattos DM (2020) Blockchain reputation-based consensus: a scalable and resilient mechanism for distributed mistrusting applications. Comput Netw 179

21. Yue K, Zhang Y, Chen Y, Li Y, Zhao L, Rong C, Chen L (2021) A survey of decentralizing applications via blockchain: the 5G and beyond perspective. IEEE Commun Surv Tutor 23(4):2191–2217. https://doi.org/10.1109/COMST.2021.3105233
22. Cruz JP, Kaji Y, Yanai N (2018) RBAC-SC: role-based access control using smart contract. IEEE Access 6:12240–12251
23. Li R, Sun Y, Liu C, Wen Y, Liu Y (2024) Distributed data sharing and access control in industrial IoT using blockchain technology. In: 2024 5th International conference on computer engineering and intelligent control (ICCEIC). IEEE, pp 372–375
24. Das S, Namasudra S (2022) Multiauthority CP-ABE-based access control model for IoT-enabled healthcare infrastructure. IEEE Trans Industr Inf 19(1):821–829
25. Luding Y, Jiahao W (2023) Decentralized fine-grained access control for edge computing leveraging Smart contracts. In: 2023 20th International computer conference on wavelet active media technology and information processing (ICCWAMTIP). IEEE, pp 1–5
26. Ding S, Cao J, Li C, Fan K, Li H (2019) A novel attribute-based access control scheme using blockchain for IoT. IEEE Access 7:38431–38441
27. Kim B, Shin W, Hwang D-Y, Kim K-H (2021) Attribute-based access control (ABAC) with decentralized identifier in the blockchain-based energy transaction platform. In: 2021 International Conference on Information Networking (ICOIN). IEEE, pp 845–848
28. Ressi D, Romanello R, Piazza C, Rossi S (2024) AI-enhanced blockchain technology: a review of advancements and opportunities. J Netw Comput Appl 225