Digital Twins Resource Management and Access Control using Blockchain, Smart Contract and Chainlink

Ralph Deters
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
ralphdeters@gmail.com

Yongchang He
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
yoh534@usask.ca

Abstract — This paper presents a novel approach for ensuring resource management and access control of digital twin resources using Ethereum blockchain, smart contracts, and Chainlink. By leveraging blockchain, smart contract and Chainlink technology, we propose a decentralized and secure approach to managing resources and controlling access to these cloud-based digital twin assets. Our approach uses Ethereum smart contracts to define the rules and conditions for accessing digital twin resources. We integrate Chainlink framework to enable the Ethereum smart contracts to access off-chain data from digital twin cloud platform. We build external adapters to empower HTTP RESTful operations towards digital twin resources and services. Digital twin management history is secured by saving the access control and operational transactions on the Ethereum blockchain. We demonstrate the effectiveness of our approach through a proof-of-concept implementation, which shows how our system can be used to manage and secure access to digital twin resources. The results show that our approach provides a scalable and secure solution for managing resource and access control in IoT applications. We conclude that blockchain and smart contract technology, combined with Chainlink, offer a promising solution for managing and securing IoT resources.

Keywords—Ethereum Blockchain, Smart Contract, Chainlink, Digital Twins, Access Control, Resource Management

I. INTRODUCTION

The Digital Twin (DT) presents one of the key concepts of Industry 4.0[1], [2]. It was generally described as a digital representation of a physical asset. In the physical world, the real entities will have the data collected through sensors. On the virtual side, digital models maintain the information about the specific asset. NASA's DT spacecraft project a decade ago made DT well-known to the Internet of Things (IoT) industry. Nowadays, the top trending technologies like Cloud Computing platforms, Big Data architectures, and the IoT are increasingly integrated, making DT solutions come true. A wide range of applications today such as smart homes, aircraft engines, wind turbines, off-shore platforms and even E-healthcare are all integrated with DT solutions.

However, managing operations and securing access to digital twin resources is a major challenge. Digital twins are complex platform that typically require fine-grained access policies to secure the related functions, models, sensitive trade secrets, and other confidential data. Weak access management and centralized policies can severely compromise the data security. Moreover, the digital twins' operation history and role-granting logs can also be easily removed due to the current centralized digital twin architecture. To address this challenge, blockchain technology has emerged as a promising solution, offering a decentralized and transparent system for managing digital twin assets. Additionally, smart contract technology provides a means of automating processes and ensuring that resources are utilized according to predetermined rules. Recently, the integration of Chainlink technology into blockchain-based systems has added new capabilities for accessing real-world data from external sources, making it possible to create more robust and secure systems.

In this work, we aim to design, develop, and evaluate a digital twin resource management and access control system using blockchain, smart contract, and Chainlink technologies. The goal of this system is to provide a secure, transparent, and decentralized framework for managing resource and securing access to digital twins. We will use a mixed-method approach, including conducting a literature review, designing, and developing the system prototype, and evaluating the system through testing. We will collect qualitative and quantitative data from system performance metrics. Part II conducted literature review; Part III describes the system architecture, including the design of the blockchain-based system and the use of smart contracts and Chainlink to ensure resource management and access control to digital twin entities. Part IV presents the system implementation and development process, including deployment of digital twins, the development of external adapters, the implementation of Chainlink node, the development of smart contracts, the use of Chainlink to access real-world data, and the deployment of smart contracts on a test network. Part V evaluates the system functionalities and overhead costs, including operational time consumption and real-world deployment costs. Finally, part VI concludes the work and part VII provides recommendations for future work.

II. LITERATURE REVIEW

A. Blockchain and Smart contract

The blockchain serves as an immutable ledger that allows transactions to occur in a decentralised manner[3]. In 2008, the blockchain system was first conceptualised and introduced by an individual or a community known as 'Satoshi Nakamoto', who released the blockchain white paper: Bitcoin: A peer-to-peer Electronic Cash System[4]. The proposed system enabled untrusted parties to execute transactions through a consensus algorithm named proof of work (PoW) without relying on a centralised validating authority[5]. As there is no single point of trust/failure, the participating peers solve a complicated problem called mining to reach a consensus with each other and vote to add a new block, including its transactions and corresponding timestamp to the chain[6]. Blocks are then chained through a hash function

reference in chronological order. Tampering with the record of transactions would alter the hush of the block, producing a failure in the chain, easily noticeable and verifiable by any node connected to the chain. Strictly speaking, all nodes share a transparent copy of the ledger with persistency. Therefore, without any central authorities, untrusted parties can share a consensus view.

The application of Bitcoin was restricted to exchanges of cryptocurrency values before the invention of the 'smart contract', coined by Nick Szabo in 1994[7], [8]. The author described the implementation of self-executing contracts in computer code, in which a protocol would enforce all the agreed terms automatically between the distributed nodes. However, programming contracts with complicated logic is not possible due to the limitations of the Bitcoin scripting language. Newly developing blockchain platforms such as Ethereum embrace the idea of executing user-defined programs on the decentralised ledger[9]. The popularity of the Ethereum platform has pushed itself into competition with all other cryptocurrencies.

The Ethereum blockchain allows users to create expressive customised smart contracts. Typically, smart contracts encapsulate a few pre-defined states and transition rules with triggers. After parties sign the smart contracts, they are deployed and fixed with a unique contract address. Cryptographic mechanisms ensure that once the smart contracts are verified and added to the blockchain, they cannot be tampered with later. They are then propagated by the Peer-to-Peer (P2P) network and confirmed by the nodes. The blockchain monitors the real-time status of smart contracts and executes the contracts once the trigger conditions have been met[10].

B. Chainlink Architectrue

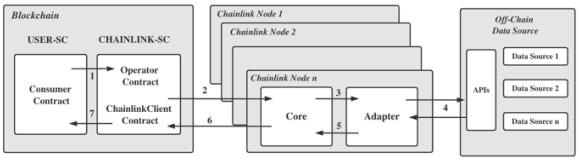
Chainlink is a decentralized oracle network that allows Ethereum smart contracts to securely access off-chain data, web APIs, and other external resources. It provides a way for smart contracts to interact with the real world by allowing them to access data from external sources[11].

The decentralized off-chain Chainlink nodes connect to the blockchain networks. These nodes independently harvest responses to off-chain requests. These responses are combined using various consensus methods to create a single, global response that is sent back to the original contract that requested the information. The Chainlink nodes are powered by a standard open-source implementation that manages blockchain interactions, scheduling, and communication with common external resources[12].

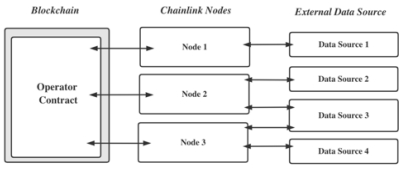
The Chainlink core algorithm is responsible for facilitating communication with blockchains, organizing tasks, and delegating work to different external services[13]. The tasks, known as jobs, assigned to Chainlink nodes are divided into smaller units called subtasks, which are processed in a specific order. Each subtask has a defined operation it performs, and the result is passed on to the next subtask until a result is obtained. Chainlink's node software includes several built-in subtasks, such as making HTTP requests, parsing data, and converting data to different blockchain formats. Chainlink currently uses a schema framework that relies on JSON Schema to indicate the required inputs for each adapter and their required formatting[14].

Node operators can expand the nodes' capabilities by installing external adapters, which enable access to certain off-chain services. Adapters are third-party services that have basic REST APIs[15]. By designing adapters in a service-oriented way, it's possible to create programs in various programming languages by simply adding small intermediate APIs to the program. This approach also makes it easier to interact with complex multi-step APIs by breaking them down into individual subtasks with specific parameters.

Figure 1 (a) describes the typical workflow of Chainlink nodes. 1) The data requesting contract, known as consumer contract (denoted as USER-SC by Chainlink team), makes an on-chain request; 2) CHAINLINK-SC, specifically, Operator contract and ChainlinkClient contract, emits an event (named OracleRequest) for the registered off-chain Chainlink oracle node(s); 3) Chainlink core in the node then picks up the event and routes the job to an adapter; 4) The adapter performs a request to an external API; 5) The adapter processes the response and passes it back to the Chainlink core; 6) Chainlink core reports the data to CHAINLINK-SC; 7) CHAINLINK-SC aggregates responses and passes them back as a single response to USER-SC. Figure 1 (b) shows the relationship among blockchain, Chainlink nodes and real-world data sources.



(a)



(b)

Fig. 1. Chainlink nodes' workflow

The Operator contract and Chainlink Client contract are both key components of the Chainlink decentralized oracle network, but they serve different roles. The Operator contract is responsible for managing the overall operation of the oracle network, including registering new oracle nodes, managing security deposits, and coordinating the delivery of data to smart contracts[16]. This contract is primarily responsible for managing the relationships between the various nodes in the network. The ChainlinkClient contract, on the other hand, is responsible for retrieving data from external data sources and delivering it to users' consumer contracts[17]. When a consumer contract needs to access data from an external source, it sends a request to the ChainlinkClient contract, which then routes the request to an appropriate Chainlink oracle node.

C. Digital Twins

A digital twin (DT) is a copy of a physical thing, produced by a computer. Real-world data is incorporated into the digital twin copy computer programs to produce simulations which will be used to anticipate how a product or process will

function in the future. As is shown in Figure 2, Using the assets' sensors physically built-in, real-time data can be gathered and mapped into a virtual model. Artificial intelligence, software analytics and the internet of things (IoT), plus advanced technologies such as machine learning and the booming availability of big data, can all be used to generate satisfactory results[18].

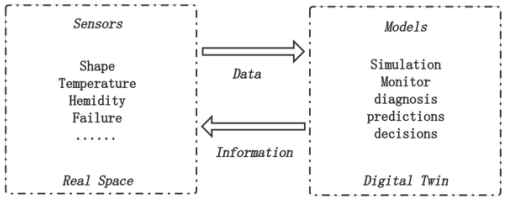


Fig. 2. Relationship of real space and digital twin

The idea of DT was first voiced in 1991, with the publication of Mirror Worlds, by David Gelernter. The use of DTs became increasingly popular during the digitization of equipment and manufacturing systems in the engineering industry in the early 2000s. General Electric (GE), for example, builds cloud-based DTs of its physical assets that analyze data collected from sensors with state-of-the-art AI, physics-based models, and data analytics to better manage its physical assets[19]. The concept of DTs can be broadly applied to many technologies and is thus likely to disrupt industries beyond manufacturing. For example, biologists have created digital twins of trees, which would be able to digitally examine their internal and external factors, and measure the amount of water consumed, oxygen released, and sunlight received. They would also determine the age of the tree and monitor its growth from seedling to adult. They can also combat disease and pests when environmental data is available[20]. There are companies operating in the DT market, such as General Electric (GE), Cisco, Ansys, Siemens, Bosch, Microsoft Azure, and so on[21]. Microsoft Azure Digital Twins draws more attention since it can be connected to the Azure cloud service, for example, Azure AI, Azure Stream Analytics, Azure Maps, Azure Storage, and Microsoft Mixed Reality, to name a few. Such IoT and AI platforms drastically reduce the complexity of building digital twin solutions by enabling functions that trigger automatic endpoint actions based on incoming information.

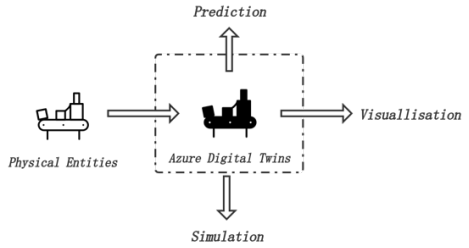


Fig. 3. Azure Digital Twins functionality

Azure Digital Twins (ADT) is a Platform as a Service (PaaS). As shown in Figure 3, ADT enables us to create and interact with real-world entities in the digital space. More generally, a DT is a virtual representation of a physical object or system, such as a car, a pump, a plant, a building, a factory or a production process. The main aim of the ADT is to create a digital copy of that physical object so that the digital entities can receive input data from sensors gathering data from the object. Because the data is a continuous stream of real-time, the digital copy is always in sync with its physical counterpart, making it possible to apply technologies, like machine learning, to the digital copies in order to produce predictions and simulations[22].

The Microsoft ADT creates the Digital Twins Definition language (DTDL), a language for describing models for IoT devices, device DTs, and logical DTs. Broadly speaking, modelling enables IoT solutions to provision, configure, monitor, and use DTs of all kinds from multiple sources in a single solution. The DTDL is made up of a set of metamodel classes that are used to define the behaviour of all DTs. Six metamodel classes are defined to describe DT behaviour: Interface, Telemetry, Property, Command, Relationship and Component.

D. Access Management

Access management is important because it allows to control who has access to certain resources or actions[23]. This is especially important in a decentralized system, where anyone can potentially interact with a smart contract[24]. Without proper access management, a contract could be vulnerable to unauthorized use or manipulation. For example, if a contract allows anyone to call a function that modifies critical data, an attacker could potentially exploit this to gain unauthorized access or make malicious changes. Proper access management helps to reduce these risks by ensuring that only authorized parties can perform certain actions within the contract.

Many Identities access and management (IAM) systems use a method: role-based access control (RBAC) to assign permissions for who can do what within specific resources. RBAC allows to create and enforce advanced access by assigning a set of permissions. The permissions are based on what level of access specific user categories require to perform their duties. In other words, different people in your framework can have completely different levels and types of access privileges based solely on factors such as their job function and responsibilities. For example, Human Resources employees could view employee records but not customer data. And an HR manager could delete or change HR records while a lower-level HR specialist would only be able to view them. When an individual's responsibilities or functions change—for example, due to a promotion or department transfer—that person is assigned to the new role in the RBAC system[25].

With the attribute-based access control (ABAC) approach, we are essentially breaking down a user—and the circumstances around their request—into attributes that define their access in your system[26]. While the system considers the user's tags when authorizing, it also considers attributes for the resource, the environment, and the action that user is trying to take. For example, if a customer support representative is trying to open and edit a contract, they might find that they can only read the file. The level of control in this system is high, but so is the effort. With attribute-based control, every access decision runs through the admin role—which means the admin role has complete control over who gets access to what at any given time.

Capability-based access control (CapBAC) is a concept in the design of secure computing systems, one of the existing security models. A capability (known in some systems as a key) is a communicable, unforgeable token of authority[27].

It refers to a value that references an object along with an associated set of access rights. A user program on a capability-based operating system must use a capability to access an object. CapBAC refers to the principle of designing user programs such that they directly share capabilities with each other according to the principle of least privilege, and to the operating system infrastructure necessary to make such transactions efficient and secure.

Blockchain technology has several advantages that make it a suitable platform for access control applications. These advantages include decentralization, immutability, transparency, the ability to use smart contracts, and a high level of security and privacy. Blockchain technology enables the creation of secure, efficient, and compliant access control mechanisms by providing a tamper-proof, transparent, and automated platform. Using smart contracts, blockchain can automate access control processes and ensure that access control policies are enforced automatically and transparently. The transparency and immutability of blockchain technology also enable auditing and compliance checks, which can be useful in regulated industries.

Abdi et al. [28] presents a review and classifications of different access control mechanisms in IoT systems. Cruz et al. [29] created RBAC-SC, which used blockchain and smart contract to realize a challenge-response authentication protocol that verifies a user's ownership of roles. Islam et al.[30] designed and implemented ABAC policies on Hyperledger Fabric blockchain to enable a distributed access control for IoT applications. Nakamura et al. [31] used a delegation graph in the smart contract based CapBAC scheme, allowing object owners to verify the ownership and validity of the capability tokens of the subjects.

E. Summary

This chapter went through the background in the connected fields of blockchain, Chainlink, digital twin, and access management. There hasn't been any work reported so far on using blockchain, smart contract, and Chainlink to create a policy enforcement point for digital twin resources, e.g., Azure digital twins.

In the following Chapters III the design and architecture, and chapter IV implementation, we will discuss in detail on how the bridge between digital twins and blockchain is built with the integration of Chainlink technology, and how access control policies are enforced inside smart contract to secure access towards the proposed framework.

III. Design and Architecture

A. System Design

The project serves as a proof-of-concept (PoC), showcasing the potential and feasibility of the proposed solution. We are to explore a prototype BaaPEP - Blockchain as a Policy Enforcement Point - to secure digital twin resource and access control using Ethereum blockchain, smart contract, and Chainlink. Every component will be designed and tailored to fulfill its specific task in the proposed framework. Ethereum smart contracts will be used to define the rules and conditions for accessing digital twin resources; Chainlink framework will be integrated to enable the Ethereum smart contracts to access off-chain data from digital twin cloud platform; external adapters will be built to empower HTTP RESTful operations towards digital twin resources and services; finally, the digital twin management history will be secured automatically as logs on the Ethereum blockchain platform.

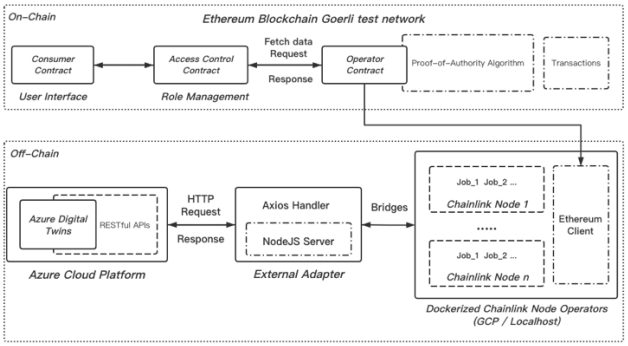


Fig. 4. Ethereum blockchain-based Digital Twin resource management architecture review

Figure 4 shows the system's components and the relationship among them. The system can be divided as On-Chain module and Off-chain module. In the On-Chain module, everything takes place on the blockchain platform. Smart contracts, including consumer contract, access control contract and operator contract, will be executed based on Proof-of-Authority algorithm; operation transactions will be logged on the blockchain permanently. In the Off-Chain module, everything happens without blockchain. Our digital twin instances live in the Azure digital twin platform, exposing RESTful APIs for the digital twins' interactive purpose. Chainlink node operators manage calls and response, forwarding data to external adapter or to On-Chain module. External Adapter translates requests between Azure cloud and Chainlink node operators. Both the two On-Chain and Off-Chain modules talk to each other via Ethereum Client, sitting inside Chainlink node operators.

B. Architecture

The functionality typically involves the following steps: system administrator or manager assign different roles and access policies to users based on specific digital twin management task. After that, a qualified user can successfully initiate a request for digital twin data in the consumer contract with the desired data parameters. The data request will be sent by the operator contract to the Chainlink node operators, which acts as a mediator between the Ethereum blockchain and the Off-Chain data source. The Chainlink node operators retrieve the data using an external adapter, which is a piece of software running on a NodeJS server that is designed to connect to and retrieve data from Azure digital twin database via HTTP RESTful APIs. After a successful data retrieval, node operators then return the data to the Chainlink operator contract, which in turn returns the data to the consumer contract. Finally, the consumer contract processes the returned data and may perform additional actions based on the data received. Detailed descriptions for each component will be demonstrated in the following sections. Figure 5 describes how the prototype works, data flows, and components interact with each other, to help deliver demonstration.

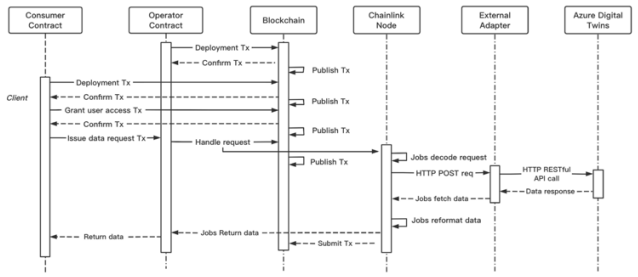


Fig. 5. Dataflow and components interaction

C. User Interface

Remix IDE is used for Ethereum smart contract Management and for user interactive purpose. It is a web-based IDE (Integrated Development Environment) that is specifically designed for writing, testing, and deploying smart contracts on the Ethereum platform. It provides a number of tools and features that make it easy to create and work with smart contracts, including: a built-in Solidity compiler for compiling and deploying smart contracts; a code editor with syntax highlighting and error checking for writing smart contract code; a debugger for identifying and fixing errors in smart contract code. MetaMask is another user interface that allows users to interact with decentralized applications (dApps) on the Ethereum network. When transaction triggered, MetaMask allows us to review and confirm the details of the transactio. Goerli testnet will be selected for the blockchain and smart contracts deployment and testing. It is a proof-of-authority (PoA) testnet, which means that it is secured by a set of trusted nodes rather than the proof-of-work (Pow) mechanism used by the main Ethereum network.

D. Access Policies Design

The core of the access policies is the concept of roles. The consumer contract will define two roles based on our usage scenario: *DEFAULT_ADMIN_ROLE, MANAGER_ROLE, and READ_ONLY_ROLE*.

The *READ_ONLY_ROLE* is a role that grants read-only access to certain functions of the contract, such as reading data from the *Azure Digital Twin*. The *MANAGER_ROLE* is a role that grants full access to all functions of the contract, including creating, reading, updating, and deleting Azure Digital Twins. The *DEFAULT_ADMIN_ROLE* is assigned to the contract creator upon deployment. This role can grant and revoke roles for managers and read-only users. We designed *grantReadOnlyRoleWithExpiration* function that allows the contract owner or a user with the *MANAGER_ROLE* to grant the *READ_ONLY_ROLE* to another user with an expiration time in hours. We also designed *grantManagerRoleWithExpiration* function, which allows the contract owner to grant the *MANAGER_ROLE* to another user with an expiration time. The *revokeManagerRole* function is designed to allow the contract owner to revoke the *MANAGER_ROLE* from a user. The *revokeGuestRole* function is designed to allow the contract owner or a user with the *MANAGER_ROLE* and not expired to revoke the *READ_ONLY_ROLE* from a user.

Another function named *renounceRole* will be designed allows a user with either the *MANAGER_ROLE* or *READ_ONLY_ROLE* to renounce their own role. Table 1 shows access policies for each role.

Table 1. Relationship among actors, roles and responsibilities

Actors	Roles	Responsibilities
Admin	*Default_admin_role*	Create, read, update, and delete adt Grant *manager_role* with expiration Grant *read_only_role* with expiration Withdraw contract balance Check contract balance Check contract owner Check who has what role Revoke *manager_role* Revoke *read_only_role*
Manager	*Manager_role*	Create, read, update adt Grant *read_only_role* with expiration Check contract balance Check contract owner Check who has what role Renounce role Revoke *read_only_role*
Guest	*Read_only_role*	Read adt Check contract owner Renounce role

E. Smart Contracts Design

Consumer contract is designed to enforce access policies described in the previous section to users based on *RBAC* protocol and to remotely manage *CRUD* (Create, Read, Update, and Delete) operations on Azure digital twins. State variables will be defined to store Azure digital twin interactive results. Functions will be implemented to interact with Azure digital twins and to enforce access policies.

The consumer contract will use several existing external contracts for its operations. It will use *ChainlinkClient.sol* to create and send data requests to a specific Chainlink node operator. *ChainlinkClient.sol* acts as a wrapper around the Chainlink service. It provides several helper functions for initiating and processing Chainlink requests.

The consumer contract will also implement *AccessControl.sol*. It provides a permissioned access control mechanism to other contracts. The purpose of the contract is to enable granular control over who can perform certain actions within a smart contract. The *AccessControl* contract allows other contracts to define their own roles and manage the membership of those roles. Contracts can create roles using the *createRole()* function and assign members to those roles using the *grantRole()* function. Members can be removed from roles using the revokeRole() function. Once roles have been defined and members assigned, other contracts can use the hasRole() function to check whether a particular address has permission to perform a certain action.

There are also functions designed to request off-chain data. They are *ADT_create, ADT_read, ADT_update, and ADT_delete*. The first thing the function does is check whether the caller (the user or contract calling the function) has the *DEFAULT_ADMIN_ROLE* or *MANAGER_ROLE* and whether the block timestamp is less than the expiration time stored in the *roleExpiration* mapping for the caller. If either of these conditions is not met, the function will throw an exception with the message Access denied or role expired.

The function then creates a new *Chainlink.Request* object called req and assigns it the result of calling the *buildOperatorRequest*. The function emits an event called *RequestMultipleFulfilled* with the given request ID as an argument, and it sets the value for state variable in consumer contract.

F. *Azure Digital Twins Design*

The azure portal site and the Azure Digital Twins Explorer were used for visualizing and interacting with ADTs data in a web browser[32].

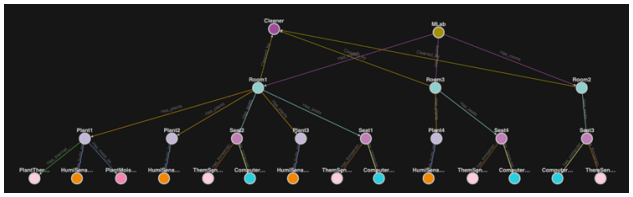


Fig. 6. Model graph: Relationship of DT interfaces

Azure digital twin models are represented in the JSON-LD-based Digital Twin Definition Language (DTDL). The twin graph in Figure 6contains three rooms, named room1, room2 and room3, respectively, and each room may have several plants and seats. A cleaner is responsible for the three rooms' cleaning job. Each plant can have three types of sensors to monitor its healthy condition.

G. *External Adapter Design*

The external adapter is a piece of software connecting the deployed Chainlink node operator, locally or remotely, to an external data source. It is used to send and fetch data to and from the Azure Digital Twin platform and return it to the Chainlink node operator, which can then use that data as input to a smart contract on the blockchain. We build Axios POST methods, wrapping the HTTP GET request, to launch request to Azure Digital Twin database, and response to any POST request from Chainlink nodes. Figure 7 shows the working mechanism.

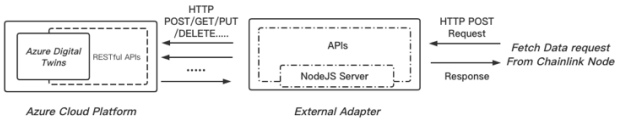


Fig. 7. External Adapter working mechanism

H. *Chainlink Node Operator Design*

We have defined four jobs inside Chainlink Node Operator to fulfill data requests from consumer contract, as shown in Figure 8. These jobs are composed of a series of tasks that are executed in a specified order. They are designed to interact with Azure Digital Twin services or APIs and to return the requested data back to the smart contract.

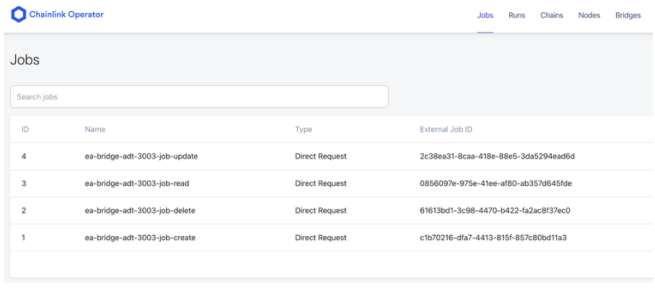


Fig. 8. Jobs defined inside a Chainlink Node Operator

Fig. 9. Job definition in a Chainlink node operator

Figure 9 demonstrates the detailed definition of the job named ea-bridge-adt-3003-job-create. The job is defined to interact with Azure Digital Twin service to create a digital twin. The job is triggered when an Ethereum event is emitted by the smart contract, and it processes the event data to fulfill the request. The job has attributes and a series of tasks to process the requests and generate the response. Tasks are defined in the attribute *observationSource*.

IV. IMPLEMENTATION

A. *Azure Digital Twin Implementation*

We have built a simulation to keep our Azure digital twins active. In this simulation, we feed simulation data to our digital twin instance. Azure digital twin is designed to work alongside IoT Hub, an Azure service for managing devices and their data. We connect the IoT Hub to the *ProcessHubToDTEvents* Azure function, so that the digital twin data can flow from the simulated device in digital twin through the function. Property change events in Azure Digital Twin platform are routed to the Event Grid topic, where the *ProcessDTRoutedData* Azure function will be used to listen for the events and set property change to digital twins. The framework is shown in the Figure 10. The running data feeding simulation screenshot is shown in Figure 11.

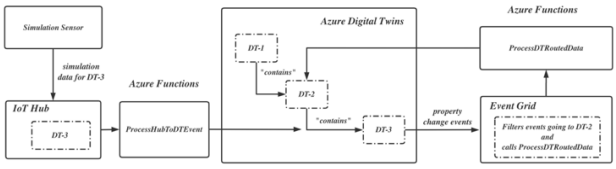


Fig. 10. Azure Digital Twin data feeding framework

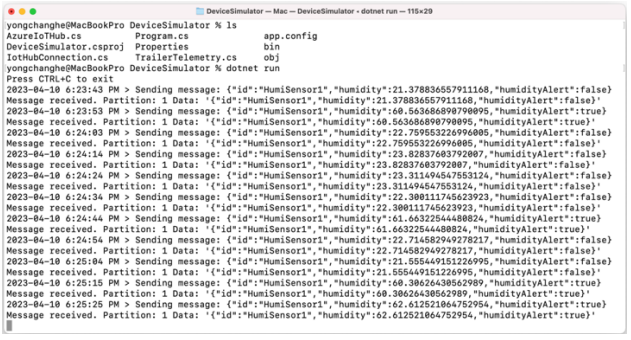


Fig. 11. Screenshot of a running data feeding simulation

B. *Chainlink Node Operator Implementation*

The Chainlink node can be deployed using docker-compoes. It can be deployed locally as localhost or remotely on any

cloud platform. If running locally, the Chainlink node operator can be accessed by visiting http://localhost:6688. In a Chainlink node, the account balance refers to the amount of Ether (ETH) that is available in the Ethereum account associated with the node operator. The account balance determines the node's ability to perform on-chain actions. To function properly, a Chainlink node needs to have a sufficient account balance to cover the cost of any on-chain actions that it performs. A job in a Chainlink node refers to a specific task. Transactions refer to the on-chain actions that the node operator performs on the Ethereum blockchain. These actions can include executing smart contracts, transferring Ether (ETH) or other tokens, and interacting with other Ethereum-based protocols. Off-chain reporting are used to securely transmit data and messages between the node and external clients.

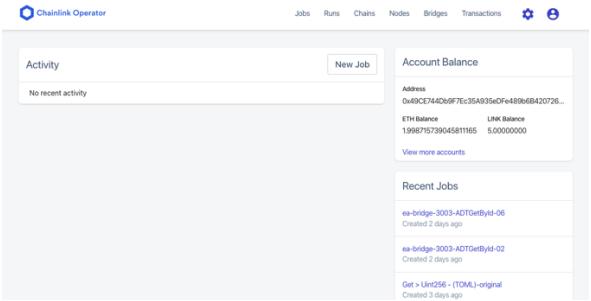


Fig. 12. Screenshot of a locally running Chainlink node operator

C. Ethereum Smart Contracts implementation

The operator contract acts as an intermediary between the consumer smart contract and the external data, enabling the consumer smart contract to securely access and incorporate data from external sources into its execution. An operator contract will include the operator functions, data retrieval and validation logic, and security measures. The Chainlink node's account address is required to be an authorized sender for Operator contract since the Chainlink node is responsible for executing certain tasks on behalf of the Operator contract. It can help to prevent unauthorized or malicious actors from accessing or manipulating the contract's data or functions. After the deployment, LINK token needs to be sent to the deployed contract address, so that the Chainlink Node Operator will have the necessary resource to fulfill the request in a timely and reliable manner[33]. A function named contractBalance can be used to check for the current token balance of ETH and LINK.

D. External Adapter Implementation

The server can be started by using command in CLI under the server.ts directory. Command yarn start will start the server. The external adapter listens to HTTP POST request from Chainlink Node operator on port 3003 and send back response from Azure Digital Twin platform to Chainlink Node operator. Nodemon is being used to automatically restart the server whenever a change is detected. Figure 13 shows the status of working External Adapter.

E. Access Polices Implementation

We use grantReadOnlyRoleWithExpiration function allows the contract owner or a user with the MANAGER_ROLE to grant the READ_ONLY_ROLE to another user with an expiration time in hours. We can also use grantManagerRoleWithExpiration function, which allows the contract owner to grant the *MANAGER_ROLE* to another user with an expiration time in hours. The revokeManagerRole function will be used to allow the contract owner to revoke the *MANAGER_ROLE* from a user. The revokeGuestRole function will be used to allow the contract owner or a user with the *MANAGER_ROLE* and not expired to revoke the *READ_ONLY_ROLE* from a user. If no policy is met, the function will throw an exception with the message "Access denied or role expired". Screenshot can be found below in Figure 14.

```
yongchanghe@Yongchangs-iMac ea-adapter-for-adt % yarn start
yarn start
yarn run v1.22.19
$ nodemon src/server.ts
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node src/server.ts`
Server listening on port 3003
 Request data received:  {
  data: { dtid: 'HumiSensor2' },
  id: '0x0856097e975e41eeaf80ab357d645fde',
  meta: {
    oracleRequest: {
      callbackAddr: '0xBD7f65c2f9470F71Cbd5D1F2a0BBeF6F926A26EB',
      callbackFunctionId: '0x05562444',
      cancelExpiration: '1681068084',
      data: '0x64647469646b48756d6953656e736f7232',
      dataVersion: '1',
      payment: '1000000000000000000',
      requestId: '0xc9931e8791dc1cd9fee1215bce9cfbae1419dd3991a323ece273f550248b7013',
      requester: '0xBD7f65c2f9470F71Cbd5D1F2a0BBeF6F926A26EB',
      specId: '0x3038353630393765393735653431656561663830616233353764363435666465'
    }
  }
}
returned response:   {
  data: {
    result: { id: 'HumiSensor2', humidity: '20', humidityAlert: false }
  },
  jobRunId: '0x0856097e975e41eeaf80ab357d645fde',
  statusCode: 200
}
```

Fig. 13. External adapter listening data request and returning response

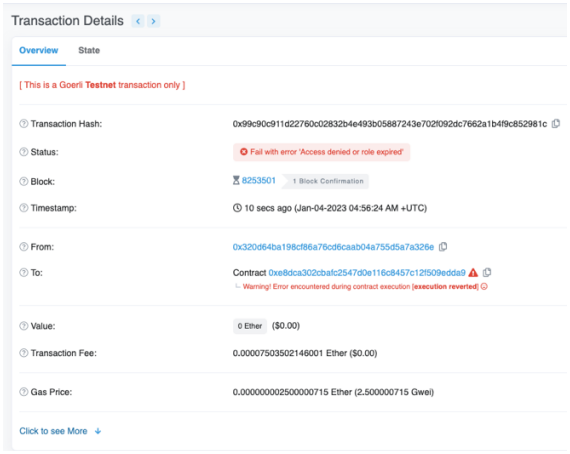


Fig. 14. Transaction fails without access privilege

F. Visiting Azure Digital Twins from Consumer Contract

We can visit Azure digital twins' data resource after every component is properly set. The digital twin data request can be issued by executing the digital twin's information in consumer contract, as is shown in Figure 15.

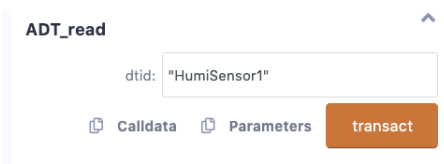


Fig. 15. Issue digital twin data request

This request will then be forwarded to the job with the same id running in the Chainlink Node Operator. The job containing a list of tasks will be executed. A screenshot is shown in Figure 16.

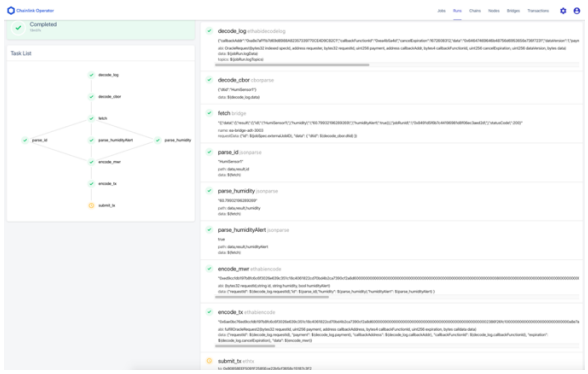


Fig. 16. Tasks in a job defined in a Chainlink Node

The running job will issue a HTTP POST request to the external adapter. The external adapter will talk with Azure digital twin's APIs and respond to this request. After that, the running job will fetch data from the response of external adapter and parse the required data. These data then will be encoded and send to consumer contract via operator contract, as is shown in Figure 17.

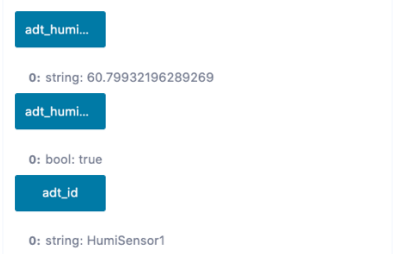


Fig. 17. Data shown in consumer contract from Azure Digital Twin

V. Evaluation

Digital twins are deployed on Azure Digital Twin platform; smart contracts are deployed on Ethereum Goerli testnet; Chainlink node and external adapters are implemented on the host machine, as is shown in Figure 6.1. The host machine is running macOS Monterey version 12.6.3. The processor is 3.2GHz Quad-Core Intel Core i5. The memory is 24 GB 1600 MHz DDR3. Execution time and costs are evaluated.

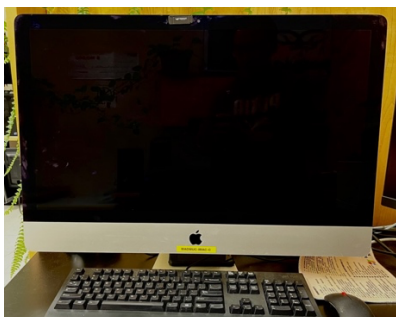


Fig. 18. Host machine implemented Chainlink node and external adapter

A. Time Consumption Evaluation

We found that the transaction time spent to deploy a contract can fluctuate more when spending less money, e.g., 1.0GWei. We can find a trend that the more we spend for making the transaction, the more likely that we spend a shorter time for the deployment. Afterall, the time consumption of a transaction can be influenced by many other factors including the network condition, number of transactions waiting to be executed and transaction complexity. Contract deployment can be finished within a minute (45 seconds).

The time consumption for Azure Digital Twins resource management is much longer. This is due to the special dataflow structure. We can notice that 50% of the CRUD operation transactions were finished within a minute (30 – 60 seconds), while some transactions can run beyond one minute (up to 89 seconds). After rearranging the time consumption distribution results in terms of different gas price we offered, it is easy to notice that the more we paid for the CRUD operation, the more likely that the transaction will be finished within a shorter period. But still, the average execution time can go around 38 - 45 seconds.

In terms of time consumption for access management, we can conclude that transactions can be finished within 20 seconds.

B. Cost Evaluation

We investigated Azure Digital Twins pricing on February 26, 2023. Azure Digital Twins uses a pay as you go pricing model that involves three main components: Operations, Messages, and Query Units[34]. Operations are the common managements for Azure Digital Twins, including creating, reading, updating, and deleting models and twins. Messages are sent by the digital twins to report changes in their properties while events are sent by external systems to trigger actions in the digital twins' instances. Query units allows users to query digital twins' properties. APIs are provided for the three types of interaction components by Azure Digital Twins platform. Users of Azure Digital Twins will only get billed for the number of Operations, Messages, and Query Units consumed during the billing period. They will not be charged if they only created digital twin instances and don't have any operations, messages and Query units. The table below is the pricing policy (on February 26, 2023, 1 USD = 1.3331 CAD). Azure for Students account can get free access to Azure Digital Twin service.

The most expensive cost is the deployment of the Ethereum smart contract: consumer contract and operator contract, as is shown in Figure 19. Theoretically, the more Gas price you offer for the deployment, the more you will pay for the transaction fee, and the less time to wait for making the transaction. However, based on our experiment we found that increasing the transaction fee did speed up several sends transaction time, but in general we still need to wait for 10 - 25 seconds to complete the transaction. So, in this stage we don't suggest investing money for boosting the contract deployment transaction time. Spending 8.43 CAD and 7.84 CAD to deploy the two Ethereum smart contracts are the most reasonable choices. The transaction fee in ETH is converted to CAD based on the live price on Coinbase[35].

The transaction fees for Azure Digital Twins resource and access management are much cheaper compared with contract deployment transactions, as are shown in Figure 20 and 21

respectively. For Azure Digital Twins resource management, the costs for CRUD one time operation are range from 0.41 to 0.76 CAD based on different Gas price offered. The transaction for Updating an Azure digital twin information cost highest (0.76 CAD) in this group. The transaction fees for user access management cost less. But it is noticeable that granting a role can cost much higher compared to revoking a role and renouncing a role.

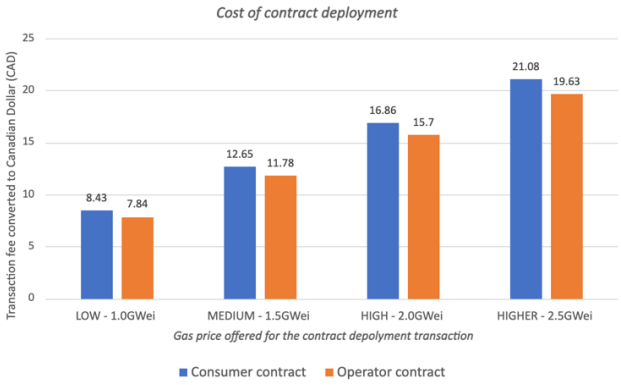


Fig. 19. Cost of contract deployment (converted to CAD)

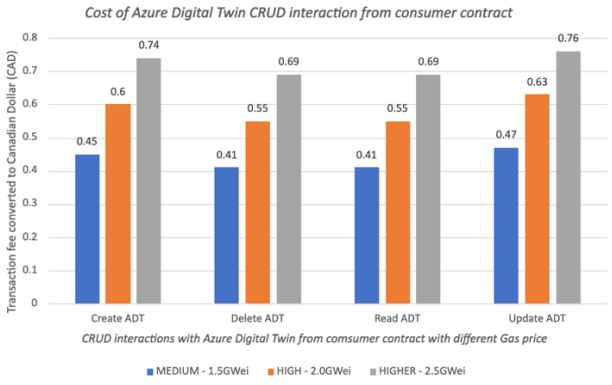


Fig. 20. Cost of Azure Digital Twin CRUD interaction from consumer contract

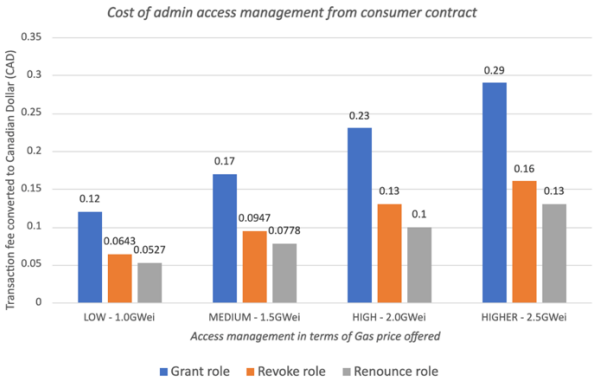


Fig. 21. Cost of admin access management from consumer contract

VI. Conclusion

This work proposes, implements, and evaluates a novel solution that combines Ethereum blockchain, smart contract, and Chainlink technology to address the digital twin resource management and access control challenges. The system allows the whitelisted users to enforce access policies and manage their digital twin resources in a decentralized manner. The approach employs Ethereum smart contracts to establish the rules and conditions for accessing digital twin resources. By integrating the Chainlink framework, our approach enables the Ethereum smart contracts to access off-chain digital twin data. Additionally, we have developed external adapters that enable RESTful operations from Ethereum smart contract towards digital twin resources and services. All access control and operational transactions are stored on the Ethereum blockchain.

VII. Future Work

The successful implementation has shown that the proposed architecture can provide a secure and transparent environment for managing digital assets and access control. However, there is still room for improvement in terms of future work in this research.

Firstly, the Ethereum's consensus algorithm, namely Proof-of-work or Proof-of-Authority, can result in relatively long transaction confirmation times and transaction fees. Looking ahead, we plan to explore the use of Hyperledger blockchain for implementing our proposed framework. Hyperledger offers faster transaction processing and low costs, making it another ideal choice for implementing our framework in future scenarios.

Secondly, it will be worth investigating how the overall performance of our proposed framework can be affected when the job is defined with an alternative method. In such approach, instead of defining a single job within a node (distributed), the job will be divided into smaller sub-tasks that can be distributed across multiple nodes in the network. These nodes can then process the sub-tasks in parallel, allowing for more efficient data processing.

Thirdly, it will be worth studying the situation where the proposed framework is applied to a real-world scenario with a certain number of users. The settings can be a group of people routinely using this framework for research purpose.

References

[1] P. Augustine, "The industry use cases for the Digital Twin idea," *Advances in Computers*, vol. 117, no. 1, pp. 79–105, Jan. 2020, doi: 10.1016/BS.ADCOM.2019.10.008.

[2] P. Raj and C. Surianarayanan, "Digital twin: The industry use cases," *Advances in Computers*, vol. 117, no. 1, pp. 285–320, Jan. 2020, doi: 10.1016/BS.ADCOM.2019.09.006.

[3] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, pp. 557–564, Sep. 2017, doi: 10.1109/BIGDATACONGRESS.2017.85.

[4] S. N.-U. https://bitcoin. org/bitcoin and undefined 2008, "Bitcoin whitepaper," *huobi-1253283450.cos.ap-beijing ...*, Accessed: Jun. 22, 2022. [Online]. Available: https://huobi-1253283450.cos.ap-beijing.myqcloud.com/1543476765952_IgOT7VVGO4Vr3QUjymBa.pdf

[5] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Čapkun, "On the security and performance

of Proof of Work blockchains," *Proceedings of the ACM Conference on Computer and Communications Security*, vol. 24-28-October-2016, pp. 3–16, Oct. 2016, doi: 10.1145/2976749.2978341.

[6] N. Islam, Y. Marinakis, S. Olson, R. White, and S. Walsh, "Is BlockChain Mining Profitable in the Long Run?," *IEEE Trans Eng Manag*, 2021, doi: 10.1109/TEM.2020.3045774.

[7] "Szabo: Smart contracts - Google Scholar." https://scholar.google.com/scholar_lookup?author=N.+Szabo+&publication_year=1994&title=Smart+Contracts (accessed Jun. 26, 2022).

[8] "Smart contracts: building blocks for digital markets - Google Scholar." https://scholar.google.ca/scholar?hl=en&as_sdt=0%2C5&q=Smart+contracts%3A+building+blocks+for+digital+markets&btnG= (accessed Jun. 26, 2022).

[9] V. B. paper and undefined 2014, "A next-generation smart contract and decentralized application platform," *nft2x.com*, Accessed: Jun. 26, 2022. [Online]. Available: https://nft2x.com/wp-content/uploads/2021/03/EthereumWP.pdf

[10] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9604 LNCS, pp. 79–94, 2016, doi: 10.1007/978-3-662-53357-4_6.

[11] "Chainlink Overview | Chainlink Documentation." https://docs.chain.link/getting-started/conceptual-overview (accessed Jan. 01, 2023).

[12] "Basic Request Model | Chainlink Documentation." https://docs.chain.link/architecture-overview/architecture-request-model/ (accessed Feb. 21, 2023).

[13] A. Beniiche, "A Study of Blockchain Oracles," Mar. 2020, doi: 10.48550/arxiv.2004.07140.

[14] "Configuring Chainlink Nodes | Chainlink Documentation." https://docs.chain.link/chainlink-nodes/v1/configuration (accessed Feb. 21, 2023).

[15] "Bridges: Adding External Adapters to Nodes | Chainlink Documentation." https://docs.chain.link/chainlink-nodes/external-adapters/node-operators (accessed Feb. 21, 2023).

[16] "chainlink/Operator.sol at develop · smartcontractkit/chainlink." https://github.com/smartcontractkit/chainlink/blob/develop/contracts/src/v0.7/Operator.sol (accessed Feb. 21, 2023).

[17] "ChainlinkClient API Reference | Chainlink Documentation." https://docs.chain.link/any-api/api-reference/ (accessed Feb. 21, 2023).

[18] "What is a digital twin? | IBM." https://www.ibm.com/topics/what-is-a-digital-twin (accessed Aug. 14, 2022).

[19] "C.J. Parris et al., "The Future for Industrial Services:... - Google Scholar." https://scholar.google.ca/scholar?hl=en&as_sdt=0%2C5&q=C.J.+Parris+et+al.%2C+%22The+Future+for+Industrial+Services%3A+The+Digital+Twin%22%2C+Infosys+Insights+newsletter%2C+pp.+42-49%2C+2016.&btnG= (accessed Aug. 14, 2022).

[20] A. el Saddik, "Digital Twins: The Convergence of Multimedia Technologies," *IEEE Multimedia*, vol. 25, no. 2, pp. 87–92, Apr. 2018, doi: 10.1109/MMUL.2018.023121167.

[21] "20 Digital Twin Technology Companies. Examples of The Best Digital Twins Solution Providers — Dashdevs." https://dashdevs.com/blog/product-owner-talks-20-digital-twins-service-companies/ (accessed Aug. 15, 2022).

[22] "Azure Digital Twins Tutorial -Getting Started With Azure Digital Twins [A Step-by-Step Guide] - YouTube." https://www.youtube.com/watch?v=YBwraf72BKI (accessed Aug. 15, 2022).

[23] R. S. Sandhu and P. Samarati, "Access Control: Principles and Practice," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40–48, 1994, doi: 10.1109/35.312842.

[24] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the internet of things," *IEEE Internet Things J*, vol. 6, no. 2, pp. 1594–1605, Apr. 2019, doi: 10.1109/JIOT.2018.2847705.

[25] R. Awischus, "Role Based Access Control with the Security Administration Manager (SAM)".

[26] K. Yang and X. Jia, "ABAC: Attribute-based access control," *SpringerBriefs in Computer Science*, vol. 0, no. 9781461478720, pp. 39–58, 2014, doi: 10.1007/978-1-4614-7873-7_3/COVER.

[27] K. Hasebe, M. Mabuchi, and A. Matsushita, "Capability-based delegation model in RBAC," *Proceedings of ACM Symposium on Access Control Models and Technologies, SACMAT*, pp. 109–118, 2010, doi: 10.1145/1809842.1809861.

[28] A. I. Abdi, F. E. Eassa, K. Jambi, K. Almarhabi, and A. S. A. M. Al-Ghamdi, "Blockchain Platforms and Access Control Classification for IoT Systems," *Symmetry 2020, Vol. 12, Page 1663*, vol. 12, no. 10, p. 1663, Oct. 2020, doi: 10.3390/SYM12101663.

[29] J. P. Cruz, Y. Kaji, and N. Yanai, "RBAC-SC: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12240–12251, Mar. 2018, doi: 10.1109/ACCESS.2018.2812844.

[30] M. A. Islam and S. Madria, "A permissioned blockchain based access control system for IOT," *Proceedings - 2019 2nd IEEE International Conference on Blockchain, Blockchain 2019*, pp. 469–476, Jul. 2019, doi: 10.1109/BLOCKCHAIN.2019.00071.

[31] Y. Nakamura, Y. Zhang, M. Sasabe, and S. Kasahara, "Capability-based access control for the internet of things: An ethereum blockchain-based scheme," *2019 IEEE Global Communications Conference, GLOBECOM 2019 - Proceedings*, Dec. 2019, doi: 10.1109/GLOBECOM38437.2019.9013321.

[32] "Quickstart - Get started with Azure Digital Twins Explorer - Azure Digital Twins | Microsoft Docs." https://docs.microsoft.com/en-us/azure/digital-twins/quickstart-azure-digital-twins-explorer (accessed Aug. 17, 2022).

[33] "Fund Your Contracts | Chainlink Documentation." https://docs.chain.link/resources/fund-your-contract/ (accessed Dec. 30, 2022).

[34] "Pricing - Digital Twins | Microsoft Azure." https://azure.microsoft.com/en-us/pricing/details/digital-twins/ (accessed Feb. 25, 2023).

[35] "ETH/CAD: Convert Ethereum to Canadian Dollar | Coinbase." https://www.coinbase.com/converter/eth/cad (accessed Feb. 26, 2023).