External Token-based Authorization of Data-driven Integrations and Service Compositions in MQTT 5

David Hästbacka
Tampere University
Tampere, Finland
ORCID: 0000-0001-8442-1248

Minh Tran
Tampere University
Tampere, Finland
ORCID: 0000-0003-4637-6081

Petri Kannisto
Tampere University
Tampere, Finland
ORCID: 0000-0002-0613-8639

Mikael Filppula
Tampere University
Tampere, Finland
ORCID: 0000-0002-5832-9859

Pal Varga
Budapest University of Technology and Economics
Budapest, Hungary
ORCID: 0000-0002-8475-3610

***Abstract*—Modern connected cyber-physical systems and their integrations to traditional information systems are increasingly dependant on data and data sharing management in their integrations. Many such systems are constantly changing and evolving their composition, often including integrations to third party (data-driven) services. This paper presents a model where a service framework, used to manage microservice configurations, is also utilized to manage access to MQTT Version 5 message topics. A proof of concept is provided demonstrating how Eclipse Arrowhead as the service management layer is capable of taking care of authentication and authorization of publish and subscribe actions to MQTT topics as individually managed data services. The study shows that JSON Web Tokens (JWT) from this service framework can be used in the MQTT Version 5 headers without violating the MQTT specification as demonstrated with HiveMQ as the message broker in the proof of concept implementation.**

***Index Terms*—authorization, authentication, third-party services, data integration, energy data services, Arrowhead framework, MQTT, JWT**

I. Introduction

The future of systems and software builds upon schemes where data is shared among organizations. In the past years, this progress received boost from microservices where the systems communicate via service calls between heterogeneous systems, relying on mutually agreed interaction patterns [1]. These interactions, called orchestrations or choreographies, are based on loose coupling and publish-subscribe communication [2], [3]. Inter-organizational choreographies cause new requirements to access control and authorization to both reach scalability and maintain security. On the other hand, access control builds the foundation of a more refined theme, namely data autonomy. In data autonomy, the goal is to not only protect systems but also control who owns the data and who can use it [4].

Modern (micro)service based systems are highly dynamic, changing and evolving their structure and composition at runtime. This may include integrations to various external systems and services, e.g., provided and managed by third parties. There is a similar need to manage access to data on shared data brokers and data infrastructures. In today's connected world it would be beneficial to even have the same technical means to manage both composition of services as well as manage access to data sources and data services. Example use cases with emerging services and data ecosystems which would benefit from such increased connectivity and simplified management are, for example, smart cities, smart grids, smart production, autonomous vehicles, and various data based services spanning across domains.

This paper proposes an external authentication and authorization model for the message brokering protocol Message Queuing Telemetry Transport Version 5 (later MQTT 5 [5]). This model consists of an external entity that controls the access rights and generates the tokens for connection with a MQTT 5 broker on a topic level. No extra information is stored in the broker, which enables easier scalability when excess memory usage from many clients are prevented and system level configurations are managed outside individual brokers. The model also proves the usability of *User Properties*, a new header introduced in MQTT 5 together with its enhanced authentication properties, without modifying any payload of the MQTT packets. The model so demonstrates the potential of including multiple external authenticators to MQTT to suit the different needs. The example model uses the Eclipse Arrowhead [6] service model for the authorization service and HiveMQ [7] as the MQTT 5 broker in the implementation. The research objective is: *Design and implement token-based authentication and authorization based on an external trusted entity for data streams.*

This study presents multiple contributions.

- It demonstrates the ability to integrate security measures into MQTT 5 communication actions.
- The model provides an example usage of JSON Web Token (JWT [8]) for authentication/authorization purposes and application into MQTT 5.

This work was supported by the projects 'Distributed Management of Electricity System' (DisMa, Academy of Finland grant no. 322676) and 'Integrated automation for distribution grid and DERs' (INGA, funded by Business Finland).

D. Hästbacka, M. Tran, P. Kannisto, M. Filppula, and P. Varga, "External Token-based Authorization of Data-driven Integrations and Service Compositions in MQTT 5", in 49th Annual Conference of the IEEE Industrial Electronics Society (IECON), 2023, in press.

49th Annual Conference of the IEEE Industrial Electronics Society (IECON 2023), Singapore - https://iecon2023.org

- The system presented here is a proof of concept demonstration for external authentication for MQTT 5, and it also suggests the integration between HiveMQ and Arrowhead thanks to the customization options of HiveMQ.
- Finally, it also demonstrates that the same management tools can be used for the service control plane and for the data plane and its data access management (as illustrated in Fig. 1).

The paper is organized as follows. Section II presents previous studies related to the topics presented in this work. Section III discusses the concept of external authentication/authorization for MQTT 5. Section IV documents our implementation using Arrowhead and HiveMQ with an example. Section V summarises the results and other observations before the conclusions in Section VI.

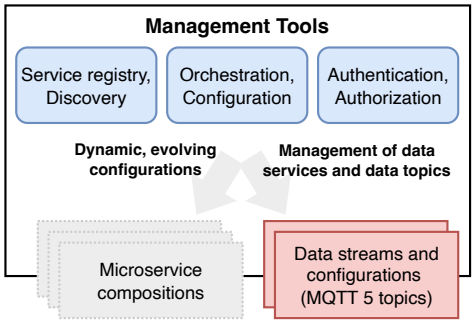

Fig. 1. Choosing the toolset for easing the burden of engineering complexity. The objective is to enable similar control tools for managing data access on a finer granularity using the same service control plane tools.

II. Related Work and Background

Data sharing necessitates interoperability, which must cover not only the information content but also trust. Effective data sharing can boost business by enabling cross-domain interoperability [9]. Interoperability and data sharing should build upon data autonomy, the ability of the data owner to control the data usage [4]. Data autonomy has resulted in joint initiatives between industries to build data spaces that are currently led by Gaia-X [10] and International Data Spaces (IDS [11]). Although data spaces have mainly targeted at controlling data autonomy regarding data sets, the autonomy concept could as well cover online data in service choreographies, including message-broker-based data exchange [12].

In data sharing, the participants must be able to authorize one another. There is a variety of authorization methods in distributed systems, based on mechanisms, such as identities, roles, attributes, or capabilities [13]. To enable trust in a distributed scheme, the authorization can occur in a "locally centralized, globally distributed" means, where a local trust service builds upon a global trust relationship [14]. Among the concrete methods, OAuth [15] is popular in commercial products. Another method, enabling distribution between organizations, is blockchain, where the participants of a network generate mutual contracts [16].

Data sharing receives benefits from service frameworks contributing to interoperability, such as Eclipse Arrowhead. Arrowhead supports creating microservice-based system-of-systems (called local automation clouds) through its service-oriented architecture. Arrowhead builds upon its mandatory core systems, namely Orchestration, Authorization, and Service Registry [6]. Still, Arrowhead as a concept is extensible with additional functionality from new modules. The examples include inter-cloud communication through gateways, event handling, protocol/data translation, plant description, service choreographing [17], and many more [18]. In the communication between separate service clouds, Arrowhead can use tokens for security and trust [19]. The overall concept keeps further evolving towards an autonomic management framework for system of systems, with industry-proven security features [20].

Despite its potential in inter-organizational data sharing, MQTT 5 standard appears poorly covered in studies for external authorization. The standard leaves degrees of freedom regarding authorization, stating that each service should implement the necessary authorization scheme [5]. The MQTT broker can enable developers to implement their own authorization system as an extension, as shown by HiveMQ [7]. Earlier, researchers have suggested token-based authorization [21] as well as blockchains with one-time passwords [22]. Additionally, MQTT has been suggested as a protocol for *Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework* (ACE-OAuth [23]). Currently, ACE-OAuth supports only Constrained Application Protocol (CoAP). Overall, it appears that there is a research gap regarding the integration of MQTT with external authorization providers. Still, such an integration would increase the applicability of MQTT in inter-organizational, inter-cloud schemes.

III. External Authentication and Authorization Model for MQTT 5

A. Use of MQTT 5 headers

One of the new features of MQTT 5 compared to its predecessor, MQTT 3.1.1, is User Property. This allows adding custom properties into the packets. In our experiment, we add 2 custom properties: `systemName` and `arrowheadToken`. `systemName` is for saving the name of the MQTT client, and `arrowheadToken` carries the JWT token for authentication/authorization. These properties are extracted from the packets for checking the validity of the packets and tokens.

B. JWT and Topics

A particular JSON web token (JWT) contains 3 parts: header, payload, and signature. The header contains the algorithm for encrypting the token and additional information about the token. The encryption algorithm in use is RSA, where a pair of private/public key is available for signing/validating the token. An extra field that contains the issuer of the token is added to the header.

TABLE I
THE CONTENT OF JWT TOKEN USED FOR EXTERNAL AUTHENTICATION AND AUTHORIZATION

JWT component	Content
Header	Encryption algorithm: RSA Token issuer
Payload	Client name: authorized MQTT client Topic URL: The URL to MQTT broker and topic Authorized action on the topic for the client
Signature	Signed using RSA algorithm Token generator has a pair of private/public key

TABLE II
THE INFORMATION STORED IN THE REGISTRY OF THE EXTERNAL AUTHENTICATION/AUTHORIZATION SERVICE

Registry field	MQTT component	Content
Providers	MQTT broker	URL link to the broker
Consumers	MQTT client	Client name Address of the client (host & port)
Services	MQTT topics	URL with the MQTT broker Distinct services for 2 different actions

The payload carries the information encoded into the JWT. In our design of JWT structure, the payload contains the access rights information: client name, MQTT topic URL, and action. The client name (`systemName`) is the name of the client that is requesting connection to HiveMQ. The topic URL (`resourceUrl`) is the address for the client to access the eligible topic. The URL is in the format {`MQTT url`}#{`topic`}. The action (`action`) is the MQTT action that the client is permitted on the topic, and it can be either `PUBLISH` or `SUBSCRIBE`.

The signature is the part that allows encryption of JWT token, which secures the authentication process. As discussed above, the signature is generated using RSA algorithm. A pair of public/private keys are available in the token generator, and the token validator has access to the public key for validating the token. Table I summarizes the content of the JWT token.

C. Managing authentication and authorization information in a service registry

An external authentication/authorization service needs to contain configurations and provide reliable service to MQTT clients and brokers. Usually, the information in a service registry includes systems and services, and the systems also consists of providers and consumers, depending on whether they distribute or receive information via services. Table II describes the data stored in such an external service registry.

Since the MQTT connections are always between a client and the MQTT broker, the providers can be always considered as the MQTT broker. The consumers are MQTT clients, and the services are equivalent to MQTT topics, since the topics can be seen to be provided by the MQTT brokers. Storing the authentication/authorization information like this requires clear definition of the allowed action on the topic from the client. To work this around, the permission to `PUBLISH` or `SUBSCRIBE` to a topic exists as 2 distinct services.

The authentication/authorization service also contains a token generator to generate tokens for the MQTT broker to further authorize the request packets by the MQTT clients. As discussed in Subsection III-B, the token generator possesses a pair of private and public keys. The generated JWT will be given to the client if the external authentication and authorization is successful. In addition, the external service will also share the public key to the MQTT broker to verify the validity of the JWT while the client request connection.

IV. IMPLEMENTATION USING ARROWHEAD FRAMEWORK AND HIVEMQ

The system architecture of the solution is presented in Fig. 3 depicting the relations between clients (publishers/subscribers) with the service framework and the message broker. The Arrowhead service framework is here used as the management mechanism and it is detailed in the Subsection IV-A. HiveMQ is the selected MQTT implementation and its extension is described in subsection IV-B. An example combining the tools is presented in Subsection IV-C.

A. Arrowhead services for configurations, authentication and authorization

In this paper, we use the Eclipse Arrowhead framework and its reference implementation as the main component to authorize and authenticate MQTT clients through the aforementioned JWT tokens. Therefore, while Arrowhead can contain modules for a multitude of other functions, we are considering only the three core systems that are mandatory for this authentication method. These are the Service Registry, Orchestration, and the Authorization system. These basically enable the secure application of the Service Oriented Architecture concepts of loose coupling, late binding, and lookup through Arrowhead-standardized services. Figure 2 depicts the basic architectural concept of Arrowhead-supported, service-oriented local automation clouds.

The Service Registry functions as the depository for the available MQTT topics. Topics can be registered as services

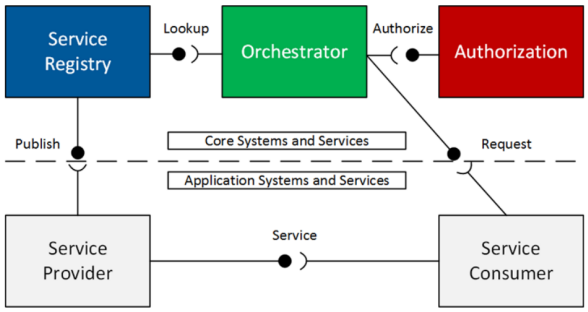

Fig. 2. The generic architectural concept of the Eclipse Arrowhead local automation cloud in the minimal setting for supporting service-oriented system of systems.

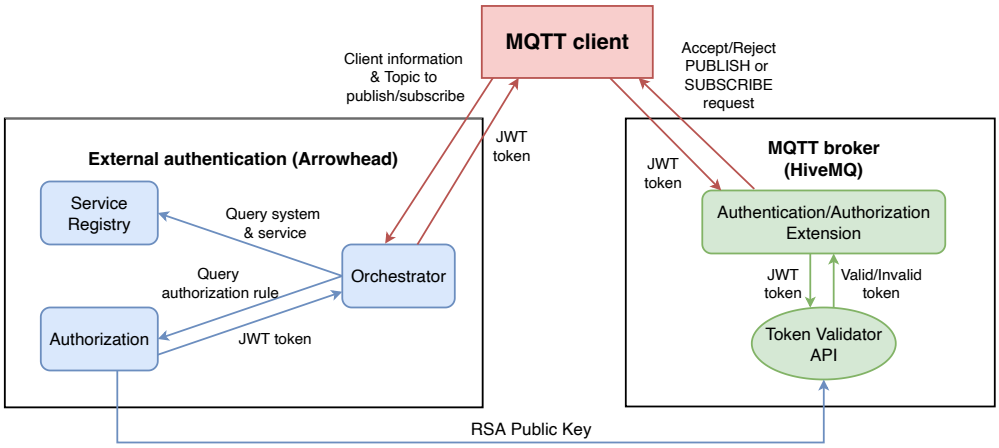

Fig. 3. Architectural diagram of HiveMQ and Arrowhead to enable token-based authentication and authorization of data streams on a topic level for dynamic and evolving data access configurations.

by providing the name of the topic, MQTT URL address where the topic can be accessed from, and the name of the service provider. Services can be queried from the registry using just the service name, but the query can be further filtered by also including other requirements, like service provider name.

The Authorization system is used to authorize any incoming service requests. Any prospecting MQTT client must be permitted access rights through the authorization before it can receive any topics through the service registry. In addition, the JWT token is generated in this module, and the MQTT service providers retrieve the public key from the module for client authentication.

Finally, the Orchestrator core system manages incoming topic requests from service consumers. Upon receiving a topic query, it will first validate the client (consumer) using the Authorization system, then retrieving an available topic from the Service Registry that matches the query and to which the client has access rights. If such a service is found, it will also give the JWT token to the client for the final authentication process with the MQTT service providers.

B. MQTT 5 extension implemented in HiveMQ

HiveMQ has a good support for extension development, providing a Java JDK with extensive documentation. Therefore, it is easy to create a custom authentication and authorization process that complies with the MQTT 5 specification.

The high-level functionalities of the extension include the decoding of the JWT token, the authentication of the CONNECT packets, and the authorization of PUBLISH and SUBSCRIBE packets. The extension authenticates the CONNECT packet by comparing the actual client that requests for a connection and the client name in the token. The authorization is performed by checking the validity of the token.

Additionally, a custom API for token validation is available in the extension. At the start of the lifetime of the extension, the extension retrieves the public key from Arrowhead and uses that public key and the issuer name in the input token to initialize the Token Validator API.

The authentication process has occurred in Arrowhead, so theoretically, no constraint on the CONNECT packet needs to be handled by the extension. However, as a safety measure, the extension still checks if the name of the client requesting connection matches the information in the token to prevent a malicious token ownership.

The extension has a similar process to validate tokens in PUBLISH and SUBSCRIBE packets. The Token Validator API provides a method to validate the token, with the topic URL and action provided as inputs. By extracting these fields in the packet, the Token Validator API can compare these fields with the content carried by the token to verify it. If the token validity is accepted by the API, the packet is accepted and the action is allowed between the client and broker. On the other hand, the client will be disconnected if it fails to provide a JWT for its request to connect, publish or subscribe.

C. Case Example for Smart Grid Data Exchange

To provide an example case for the system, let us consider the delivery of solar-panel-related data as well as the related authorization. The panel publishes two types of data: the prevailing electricity production with a regular interval and condition monitoring data for the maintenance service staff. It has been decided that the production information is available to any authenticated actor for an open electricity market. In contrast, the condition information is visible only to the owner and maintenance service provider. Therefore, both data types would have a dedicated topic with the respective subscribe permissions. On the other hand, only the software of the solar panel has the publish permission to the topics.

Whenever a client tries to either publish or subscribe to a topic, a certain authorization sequence occurs (see Fig. 4). The first interaction on the top, the MQTT broker retrieving the public key of Arrowhead, has occurred earlier in the background. The public key enables the broker to ensure that

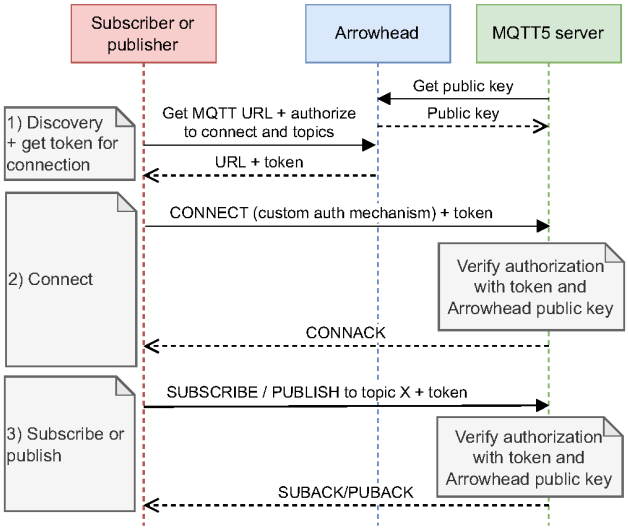

Fig. 4. The general sequence of publisher and subscriber clients communicating with Arrowhead service framework and the MQTT 5 message broker, the broker being capable of authorizing client interactions with the help of Arrowhead.

the authorizations it receives are truly from the Arrowhead instance. As soon as the client joins the network, wanting to publish or subscribe, it authenticates and receives an authorization token from Arrowhead as well as the URL of the broker. Each client, i.e., the solar panel, the maintenance service provider, and any other data receiver must do this. Next, it connects to the broker (with the CONNECT packet) and provides its token that enables the broker to decode the token with the public key, verifying the correct ownership of the token. After the verification, the server establishes a connection and replies with CONNACK. Finally, as the client wants to publish or subscribe, it sends the respective MQTT message with the same token. The MQTT broker will decode the token again to check if the access and action on the topic is valid, and when the authorization as well as the requested action succeed, the server replies with PUBACK or SUBACK. This authorization enables the broker to determine who can publish to each topic, i.e., only the solar panel. Respectively, only the maintenance service provider can subscribe for condition information, whereas any authenticated client can receive electricity production data. On the other hand, any other topics, outside the scope of this example, have their dedicated permissions.

V. Results Evaluation and Discussion

MQTT 5 introduces new enhanced features that enable the transmission of additional authentication and authorization data from both publishers and subscribers when communicating with the broker. This enables, as shown in this paper, the inclusion of JWT token data in the header to be used for managing data access on much finer levels of granularity compared to the traditional way of only authorizing access to the entire broker, which is most commonly the case with the predecessor MQTT 3.

Moreover, the approach enables an external authentication and authorization entity and the relocation of data access control to a common plane, used similar to orchestrating and evolving the microservice compositions. This enables the rapid reconfiguration of data stream configurations required by many cyber-physical systems and data-driven services connected across domains. Secondly, using an external control plane is also expected beneficial if the data infrastructure consists of multiple data broker nodes.

The tools used to validate the concept, Eclipse Arrowhead and HiveMQ, were demonstrated compatible given that an authorization module extension capable of talking to the service framework had to be developed for HiveMQ. Despite the extension or the proposed concept, it does not break the MQTT 5 specification and all communication is according the the standard.

The proof of concept also shows that a service framework can be used to manage data access as easily. Using the same tools is believed to be a major benefit simplifying configurations when integrating different systems although such benefits have not been specifically measured.

The available implementation is still a simplified version of the core systems and services of Arrowhead. The basic functionalities work well enough to demonstrate the concept of external authentication/authorization for MQTT 5. The published implementation of Arrowhead is more complex including other functionalities too. A future improvement could be the integration of the proposed concept into Arrowhead.

The system has been tested to work well when the MQTT clients are hosted locally. There has been no test to replicate the real scenario where different clients might be hosted in different clouds and request authentication from Arrowhead simultaneously in larger setups. Therefore, scalability is another point to focus on, especially this can be achieved along with the integration into Arrowhead. The approach is expected to integrate well also to resource-constrained devices as most of the the additional burden is on the broker but this has not yet been experimentally verified.

VI. Conclusion

The study highlights the increasing dependence on data and data sharing management in connected cyber-physical systems and their integrations to various data services. This paper presents a model where a service framework is used to manage authentication and data authorizations to MQTT Version 5 message topics. The model is demonstrated and validated through integration of the Eclipse Arrowhead microservice framework with HiveMQ as the message broker.

The use of JWT tokens in MQTT Version 5 headers to authenticate and authorize MQTT client requests is shown to be compatible with the MQTT specification. This decouples access control management from the message broker(s) and enables the use of an external management layer similar to how services are loosely coupled and configured at run-time.

The use of the Eclipse Arrowhead framework and HiveMQ in the implementation further validates the compatibility and effectiveness of the proposed model. However, the implementation represents a simplified version of the core systems and services of Arrowhead, leaving room for future improvements and integration of the proposed concept into Arrowhead.

Further research can focus on enhancing scalability when multiple clients hosted in different environments request authentication simultaneously. Additionally, the integration of the proposed model into Eclipse Arrowhead can be explored to leverage its comprehensive functionalities and provide a more robust solution for managing data access in connected systems.

REFERENCES

[1] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting microservices: Practical opportunities and challenges," *Journal of Computer Information Systems*, vol. 60, no. 5, pp. 428–436, 2020.

[2] D. Hästbacka, J. Halme, L. Barna, H. Hoikka, H. Pettinen, M. Larranaga, M. Björkbom, H. Mesiä, A. Jaatinen, and M. Elo, "Dynamic edge and cloud service integration for industrial IoT and production monitoring applications of industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 498–508, 2022.

[3] P. Kannisto, D. Hästbacka, T. Gutiérrez, O. Suominen, M. Vilkko, and P. Craamer, "Plant-wide interoperability and decoupled, data-driven process control with message bus communication," *Journal of Industrial Information Integration*, vol. 26, p. 100253, 2022. [Online]. Available: https://doi.org/10.1016/j.jii.2021.100253

[4] C. Fracassi and W. Magnuson, "Data autonomy," *Vanderbilt Law Review*, vol. 74, no. 2, pp. 327–383, 2021. [Online]. Available: https://scholarship.law.tamu.edu/facscholar/1457

[5] "MQTT version 5.0," OASIS, 2019, URL https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html [Visited 27 Apr 2023].

[6] P. Varga, F. Blomstedt, L. L. Ferreira, J. Eliasson, M. Johansson, J. Delsing, and I. M. de Soria, "Making system of systems interoperable – the core components of the Arrowhead framework," *Journal of Network and Computer Applications*, vol. 81, pp. 85–95, 2017.

[7] "HiveMQ," URL https://www.hivemq.com/ [Visited 27 Apr 2023].

[8] "RFC 9200 – JSON web token (JWT)," Internet Engineering Task Force, 2015.

[9] M. Magas and D. Kiritsis, "Industry commons: an ecosystem approach to horizontal enablers for sustainable cross-domain industrial innovation (a positioning paper)," *International Journal of Production Research*, vol. 60, no. 2, pp. 479–492, 2022.

[10] H. Tardieu, "Role of Gaia-X in the European data space ecosystem," in *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*, B. Otto, M. ten Hompel, and S. Wrobel, Eds. Cham: Springer International Publishing, 2022, pp. 41–59.

[11] H. Pettenpohl, M. Spiekermann, and J. R. Both, "International data spaces in a nutshell," in *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*, B. Otto, M. ten Hompel, and S. Wrobel, Eds. Cham: Springer International Publishing, 2022, pp. 29–40.

[12] P. Kannisto and D. Hästbacka, "Data autonomy in message brokers in edge and cloud for mobile machinery: Requirements and technology survey," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022.

[13] A. Khan, A. Ahmad, M. Ahmed, J. Sessa, and M. Anisetti, "Authorization schemes for internet of things: requirements, weaknesses, future challenges and trends," *Complex & Intelligent Systems*, vol. 8, pp. 3919–3941, 2022.

[14] H. Kim and E. A. Lee, "Authentication and authorization for the internet of things," *IT Professional*, vol. 19, no. 5, pp. 27–33, 2017.

[15] "RFC 6749 – the OAuth 2.0 authorization framework," Internet Engineering Task Force, 2012.

[16] P. Mell, J. Dray, and J. M. Shook, "Smart contract federated identity management without third party authentication services," in *Open Identity Summit 2019, OID 2019, Garmisch-Partenkirchen, Germany, March 28-29*, 2019, pp. 37–48.

[17] D. Kozma, P. Varga, and F. Larrinaga, "Dynamic multilevel workflow management concept for industrial IoT systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1354–1366, 2021.

[18] D. Kozma, P. Varga, and G. Soós, "Supporting digital production, product lifecycle and supply chain management in industry 4.0 by the arrowhead framework–a survey," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 126–131.

[19] C. Hegedus, P. Varga, and A. Frankó, "Secure and trusted inter-cloud communications in the Arrowhead framework," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018, pp. 755–760.

[20] S. Maksuti, M. Zsilak, M. Tauber, and J. Delsing, "Security and autonomic management in system of systems," *Infocommunications Journal*, vol. 13, no. 3, pp. 66–75, 2021.

[21] M. Calabretta, R. Pecori, M. Vecchio, and L. Veltri, "MQTT-auth: a token-based solution to endow MQTT with authentication and authorization capabilities," *Journal of Communications Software and Systems*, vol. 14, no. 4, pp. 320–331, 10 2018.

[22] F. Buccafurri, V. De Angelis, and R. Nardone, "Securing MQTT by blockchain-based OTP authentication," *Sensors*, vol. 20, no. 7, p. 2002, Apr 2020.

[23] "RFC 7519 – authentication and authorization for constrained environments using the OAuth 2.0 framework (ACE-OAuth)," Internet Engineering Task Force, 2022.