Access control for interoperable energy management systems using Verifiable Credentials

Nikos Fotiou, Spiros Chadoulos, Iordanis Koutsopoulos, Vasilios A. Siris, George C. Polyzos
Mobile Multimedia Laboratory
Department of Informatics, School of Information Sciences and Technology
Athens University of Economics and Business, Greece
{fotiou,spiroscha,jordan,vsiris,polyzos}@aueb.gr

***Abstract*—Emerging energy management systems (EMS) involve devices and services provided by multiple stakeholders. In order to improve the interoperability of these systems, state of the art efforts propose an interoperability middleware that mediates the communication between end-user applications and EMS components. The potential lack of trust between the different stakeholders raises the need for fine-grained access control mechanisms. However, extending the middleware to support access control in a secure and usable way is a challenging problem. In this paper, we present a solution that achieves fine-grained authorization using Verifiable Credentials (VCs). Our solution leverages VC properties to enable end-users to combine authorizations issued by different entities. Additionally, our solution integrates a cloud-based VC wallet that hides the authorization process from end-user applications, thus facilitating interoperability among EMSes and the development of new, secure applications.**

***Index Terms*—Authentication, Authorization, Wallet**

I. Introduction

Energy Management Systems (EMS) incorporate multiple devices and services, often provided by different providers. All these components need to exchange data in a standardized manner, while at the same time providing data protection. Indeed, energy consumers may own IoT devices from different vendors, while they are customers of different electricity retailers. Interoperability needs to be achieved, both at the data semantics level (e.g., by utilizing specific ontologies and data models), as well as at the data security level, in order to seamlessly facilitate data sharing and communication regardless of the specific providers and hardware installed [1].

In some cases, an EMS should collect data from different data providers in order to present meaningful insights to end-users, while also being able to actuate devices from different vendors which are connected to different back-ends. For example, a residential user may have installed a set of smart plugs from an IoT service provider, and an indoor climate sensor from a different provider, while a household smart energy meter is also installed by an energy retailer. In this case, the data collected from the household are transmitted by three different back-end systems, which should be semantically interoperable in order to facilitate a holistic smart home infrastructure for the end-user, e.g. through a unified mobile app interface. For that reason, many recent efforts propose an interoperability middleware that acts as a semantic "glue" between the different systems using standardized data models and ontologies [2], [3]. This interoperability middleware is responsible for facilitating data exchanges between end-user client applications (e.g. a smart-home mobile app) and the different data provider systems.

Hence, security and privacy, as well as access control schemes, are necessary for such a system since multiple providers are involved that might not fully trust each other. Specifically, smart home data generated by IoT service providers and retailers are very sensitive from a privacy perspective, since they reveal the everyday habits of the occupants, including their real-time presence.

However, adding support for access control in an interoperability middleware is a challenging problem. Such an access control solution should not rely on the middleware to store secret information; instead, it should facilitate the introduction (or the removal) of providers, and it should not hinder the development of new end-user client applications. In this paper, we provide a fine-grained access control solution by leveraging Verifiable Credentials [4], which is currently a W3C Recommendation.

A *Verifiable Credential (VC)* provides a cryptographically secure and machine-verifiable means for expressing *claims* about a *subject*. VCs are stored in user-controlled wallets and can be used (among other things) for proving possession of certain claims to a web-based verifier. Additionally, many VCs can be combined in a single *presentation*, satisfying in this way advanced authorization requirements. By leveraging these properties of VCs, in our paper, we are making the following contributions: (i) we propose an access control solution scheme that allows users to combine authorizations from multiple providers, (ii) we make our access control solution transparent both to the end-user client applications, as well as to the protected endpoints, (iii) we leverage emerging standards to manage the lifecycle of the generated authorizations, and (iv) implement and evaluate our solution.

The remainder of our paper is organized as follows. In section 2 we introduce VCs, we outline VC management protocols, currently under development by the OpenID foundation, and we discuss related work in this area. In section 3, we present the main entities considered by our solution and their interactions. In section 4, we detail the design of our system.

We present our implementation and its properties in section 5. Finally, we provide our conclusions and related discussion, as well directions for future work in section 6.

II. Background and related work

A. Verifiable Credentials

A Verifiable Credential (VC) is a W3C recommendation that allows an issuer to assert some claims about an entity, referred to as the VC subject. A VC includes information about the issuer, the subject, the asserted claims, as well as possible constraints (e.g., expiration date) [4]. This information is encoded in a machine readable format (e.g., as a JSON object in our system). A VC is usually stored in a secure enclave, referred to as the wallet. Then, a VC holder (usually, the VC subject itself) can prove to a verifier that it owns one or more VCs with certain characteristics. This is achieved by binding VCs to a holder controlled identifier (e.g., our system includes in VCs a public key owned by the holder), which enables the holder to generate a *Verifiable Presentation* (VP) of the VC(s), i.e., an object that includes one or more VCs signed in a way that can be verified using the bound identifier. VP verification does not require communication with the issuer. The VC data model allows different VC types, which define the attributes a VC should include. This provides great flexibility, since VC integrators can define their own types that fit the purposes of their systems.

Once issued, a VC can be used multiple times with different verifiers, and for a long period. For this reason it is important for an issuer to be able to revoke a VC and similarly for a verifier to be able to determine the revocation status of a VC. Our system uses for this purpose the approach specified in a recent W3C draft [5]. In particular, and in order to support revocation, an issuer maintains a revocation list that covers all *not expired* VCs it has issued. This list is a simple bit string, and each credential is associated with a position in the list. Additionally, each VC includes a property named *credentialStatus* that specifies the position of that credential in the revocation list, as well as a URL that can be used for retrieving the revocation lists. A VC is simply revoked by setting the corresponding bit in the revocation list to 1. Since the list includes only non-expired VCs, its size is tolerable for most use cases.

B. OpenID for Verifiable Credentials

The W3C recommendation specifies only the VC data model, and it does not define protocols for managing VC lifecycle. For this reason, other organizations are pursuing the specification of related protocols. The most notable effort is this of the OpenID foundation. In particular, the AB/Connect working group, has drafted two specifications: OpenID for Verifiable Credential Issuance [6], which proposes an API and corresponding OAuth 2.0-based authorization mechanisms for issuance of VCs, and OpenID for Verifiable Presentations [7], which defines a mechanism on top of OAuth 2.0 to allow the presentation of claims in the form of VPs. In the following subsections, we detail these two protocols, presenting however only the flows used by our solution.

1) OpenID for VC issuance (OID4VCI): OID4VCI enables the issuance of a VC to a wallet. OID4VCI, as used in our solution, involves the following steps :[1]

a) Issuer configuration: Initially, a user interacts with an issuer and and makes all necessary preparations required for issuing a VC (e.g., a user authenticates to a web interface of the issuer and selects and option for generating a VC).

b) Credential offer: The issuer redirects the user to their wallet by including in the redirection URL a *credential_offer* JSON object. In our system, this object includes the following parameters:

- *credential_issuer*: The URL of the issuer.
- *credentials*: A string indicating the type of the credential this offer concerns,
- *grants*: A grant that the wallet should use in the following step in order to request a token. In our system this is a key-object pair: the value of the key is "urn:ietf:params:oauth:grant-type:pre-authorized_code" and the object is a JSON object that includes a *pre-authorized_code*, i.e., a code that will be used as the grant for requesting a token:

c) Token request: The wallet requests an access token from the issuer that will be exchanged with a VC in the next step. In its request, it includes the pre-authorized code received with the previous step. The issuer responds with an *access token*, as well as with a random number referred to as *c_nonce*.

d) Credential request: As a final step, the wallet sends a credential request to the issuer. The request includes the HTTP *Authorization* header, whose value is set to *Bearer* followed by the access token received in the previous step. A credential requests includes a JSON object with the following parameters:

- format: The encoding of the requested credential. In our system the value of this paramter is set to "jwt_vc_json"
- proof: A JSON Web Signature [8] generated using a key owned by the wallet. The provided JSON Web Signature must include (among other parameters) the corresponding public key, as well as the *c_nonce* generated by the issuer in the previous step.

The issuer verifies the signature of the proof and extracts the provided pubic key. Then, it issues a VC *bound to that key*.

2) OpenID for VC presentation (OID4VP): OID4VP enables a verifier to request and receive a verifiable presentation (VP) form a wallet. OID4VP, as used in our solution, involves the following steps:[2]

a) Authorization request: A verifier requests from a wallet a VP. An authorization request includes the following parameters:

[1]It is highlighted that this section presents only the OID4VCI parameters used by our solution. For all available options of OID4VCI interested readers are referred to [6].

[2]Similarly to OID4VCI, it is highlighted that this section presents only the OID4VP parameters used by our solution. For all available options of OID4VP interested readers are referred to [7].

- *scope*: A pre-configured string indicating the type of credential(s) the VP must include.
- *nonce*: A random number that must be included in the VP signature.
- *response_mode*: Its value is *direct_post* and it indicates that the wallet must send the requested VP using HTTP post.
- *response_uri*: The (HTTPS) URI in which the wallet must post the generated VP
- *state*: An opaque value used by the verifier to maintain state between the authorization request the VP submission.

b) Response: A wallet's response to an authorization request is POSTed to the *response_uri* and includes the following parameters:

- *vp_token*: The generated VP.
- *state*: The state parameter included in the authorization request.

C. Related work

A growing number of related research efforts leverage blockchain technology in order to achieve interoperability at the security level (e.g., [9]–[11]). Nevertheless, the use of this technology usually introduces communication overhead and perhaps monetary costs. Furthermore, storing information related to access control in a (public) blockchain system may introduce security and privacy concerns. Our solution has minimal communication overhead and it does not require the storage of sensitive information by third parties.

Our system leverages VCs for expressing authorizations, but alternative mechanisms can be considered. For example, Macaroons [12], Authorization Capabilities for Linked Data (ZCAP-LD) [13], or even capabilities as defined by the WAVE framework [14] could be used instead. The advantages of VCs compared to these solutions are that the VC data model is a standardized W3C recommendation, hence it offers better interoperability and it is supported by a wide range of implementations, and VCs can be easily extended and adapted to the requirements of specific use cases, e.g., by creating a new VC *type*. Similarly, our solution uses OpenID for VC for managing the lifecycle of VCs, but other protocols can be considered, e.g., the Credential Handler API [15] or DIDComm Messaging [16]. The main advantage of OpenID for VCs is that it is based on the omnipresent OpenID Connect, therefore existing user management systems can be easily extended to integrate our solution.

Our system resembles a capabilities-based access control (CapBAC) system (e.g., [17], [18]). The main differences of our solution compared to these systems are: first, our solution uses VCs to encode capabilities, as opposed to a custom capability encoding format, second, it supports revocation, and third, it allows users to "present" to a verifier capabilities asserted by multiples issuers.

Our system encodes in a VC simple authorization decisions (e.g., "a user X is allowed to access household D"), nevertheless our design does not prevent other types of VCs that could enable more advanced access control decisions; for instance we can envision "situational oracles" (as introduced by [19]) to provide users with short-lived VCs that "describe" user-context, enabling this way context-aware access control.

The work presented in this paper, extends our previous work published in [20], [21] in such a way in order to make it applicable to the smart grid domain. In particular, the solution presented in this paper considers multiple VC issuers, performs access control by combining multiple VCs in a single presentation, defines a new VC type, and it leverages OpenID specifications to implement a cloud-based wallet that hides the authorization process from the client application.

III. Solution overview

Our system assumes an interoperable EMS, similar to [22], which is composed of the following entities.

a) End-user: this is the energy manager of the household, which wants to minimize energy costs and carbon footprint while preserving occupant comfort. Any household occupant can act as an end-user by interacting with the smart home through a *client application*.

b) EMS component provider: this can be an energy utility or an IoT service/device provider. An energy utility is the entity responsible for providing energy to the household based on a specific contract, with either static or dynamic energy prices. The energy utility can also monitor the total energy consumption of the household through a smart meter installed by them or by the Distribution System Operator (DSO). Most energy utilities also provide proprietary software to their clients so as to monitor their consumption in real-time through the smart meter, pay their electricity bills, engage consumers in energy efficiency and Demand Response (DR) schemes, etc. An IoT service/device provider is an entity that provides IoT devices, such as smart plugs, climate sensors, and smart devices, to the household, along with a proprietary platform to monitor/control them. Most smart homes incorporate multiple devices from different providers and vendors.

c) Interoperability middleware provider: it is a trusted entity that develops, deploys, and manages the interoperability middleware responsible for interoperable data exchanges and ensuring that the involved stakeholders utilize standardized data models and semantic ontologies.

Each household is identified by one or more EMS component provider-specific identifiers. Moreover, each EMS component is uniquely identified by an EMS-specific identifier. The interoperability middleware may connect to multiple EMS component providers. Figure 1 is an example of an instance of the interoperable EMS considered by our solution. As it can be seen, the interoperability middleware interacts with multiple EMS component providers. Additionally, a household may integrate components from multiple components providers. Finally, component providers may use different identifiers to refer to the same household.

The interoperability middleware exposes an API interface that converts the original APIs of the EMS component to a standardized format using specific ontologies and data models.

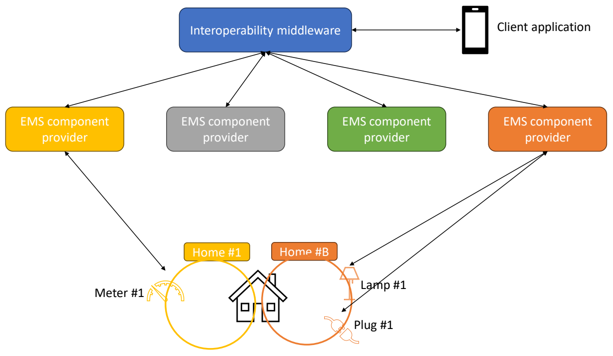

Fig. 1. The considered EMS architecture

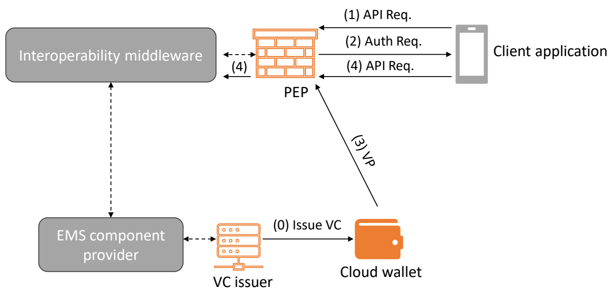

Fig. 2. The elements of our solution.

This API allows client applications to: (i) list all EMS components installed in a household (e.g., all available devices) and (ii) interact with a specific component (e.g., retrieve measurement data for a specific component in a time series form or actuate it).

Our solution endows the EMS with access control properties. In particular, we extend the EMS component provider with VC issuing capabilities, which are used for authorizing client applications to access the components installed in a specific household. Additionally, we provide a transparent proxy that acts as a *Policy Enforcement Point* and intercepts API request towards the interoperability middleware "blocking" the unauthorized ones. Finally, we implement a cloud-based wallet where issued VCs are stored. The additions of our solution are illustrated in Fig. 2. From a high level perspective access control is implemented in our solution as follows (see also the numbered arrows in Fig. 2). Initially, EMS component providers issue VCs to end-users that include the identifiers of the households they are authorized to access. An end-user performs an API request through a client application. The request is intercepted by the PEP which responds with an *authorization request* (as described in section II-B). The client application invokes the end-user wallet, which sends the appropriate VP to the PEP. From this point on, the client application is considered to be authorized and its API requests are forwarded to the interoperability middleware.

IV. System Design

Our system assumes that each VC issuer (implemented by an EMS component provider) is uniquely identified by a URL and owns a public-private key pair, which is used for signing the issued VCs. Moreover, each end-user has established an authentication method with one or more EMS component providers. Accordingly, EMS component providers provide a web interface to their VC issuer where end-users can login and request a VC.

A PEP is configured with a data structure, referred to as the *Authorization Table* where information related to the active sessions is stored. In particular, an authorization table contains tuples that map a session identifier to an authorization status, as well as to a list of household identifiers that can be accessed in the context of the corresponding session.

Our system introduces a cloud-based wallet where issued VCs are stored. Such a wallet can be self-hosted or provided as shared service by a corresponding wallet provider. In any case, end-users are assumed to have established a secure authentication method with their wallets. Similarly, wallets are assumed to be secured, i.e., they do not share user VCs without user consent and they have applied appropriate security mechanisms for protecting the private keys generated during a credential request. Finally, in the following when we say that end-users interact with their wallets it is assumed that end-user authentication has already taken place. Our solution includes five phases: (1) Set up, (2) VC issuance, (3) Unauthorized API request, (4) Authorization, and (5) Authorized API request. These phases are described in the sequel.

A. Set up

During this phase the elements of our solution are configured with the necessary information required for implementing all operations securely. The VC issuer is configured with the appropriate access control policies. Similarly, the PEP is configured with a list of VC issuers URLs and their corresponding public keys. As we discuss in the following section, our system defines a new VC type named *OwnershipCredential*; accordingly, the wallet is configured with a scope identifier named *Ownership*: if a verifier uses this scope identifier in an authorization request, the wallet "undestands" that the verifies wishes to receive a VP that includes VCs of type *OwnershipCredential*.

B. VC issuance

VC issuance in our solution is implemented using OID4VCI, and the steps described in section II-B. For the VC preparation, end-users authenticate to the web interface of the VC issuer of the EMS component providers from which they wish to receives a VC. The corresponding *credential offers* are presented as links, which when clicked redirect end-users to their wallets. The *credentials* parameter of the each credential offer is agreed to have value "OwnershipCredential".

The first time a wallet sends a *credential request* on behalf of an end-user, the wallet creates a fresh public-private key pair: the private key is then used for signing the corresponding *proof* (included in the credential request). For all subsequent requests for the same credential type (on behalf of the same end-user), the wallet uses the same public-private key pair.

Therefore, all VCs issued in our system for the same user are bound to the same public key.

Our solution defines a new type of VC named *Ownership-Credential* that includes the household identifiers that the VC subject can access. VCs in our solution are encoded as JSON Web Tokens [23] protected using a JSON Web Signature [8] generated by the issuer (following the procedure described in section 6.3.1 of [4]). Furthermore, a VC includes information that can be used for verifying its revocation status, as described in section II-A.

C. Unauthorized API request

Initially, the client application sends an unauthorized HTTPS request. The request is received by the PEP, which acts as transparent proxy. The PEP generates a new session identifier, it inserts it in the authorization table and sets corresponding entry in the authorization table as *Unauthorized*. Then it responds with an OID4VP *authorization request* (for more details see section II-B). The authorization request includes the *state* parameter, whose value equals to the session identifier.

D. Authorization

Upon receiving the authorization request the client application "invokes" the cloud wallet, providing as input the received request.[3] The wallet extracts from the authorization request the *scope* parameter and checks if it equals to *Ownership*, then it presents to the user all VCs of type *OwnershipCredential* and asks from the end-user permission to generate a VP: if the end-user consents the wallet POSTs the VP to the PEP (acting as a VC verifier). A VP in our system is a JSON Web Token signed by the wallet using the key to which the included VCs are bound. Upon receiving a VP, the PEP executes the following verifications:

1) It verifies that the VP has not expired and it is within its validity period.
2) It verifies that the *aud* claim equals to the URL of the PEP.
3) It extracts the included VCs. For each of them it verifies that: (a)it has not expired and it is within its validity period, (ii) it has not been revoked, (iii) its signature is valid and it has been generated by a trusted issuer, and (iv) it includes the same public key in the *cnf* claim.
4) Using the public key included in the *cnf* claim of the VCs it verifies the signature of the VP.

If all verifications are successful, the PEP locates the session identifier included in the *state* parameter of the posted response and updates the corresponding entry of the authorization table by setting *status* equal to *Authorized* and list of household identifiers equal to the identifiers included in the *households* claim of the POSTed VCs.

[3]Wallet invocation can be implemented using different mechanism, some of which are discussed in [7]; the mechanism used in our system is described in section V.

E. Authorized API request

As this point a client application can perform an authorized request. This is simply done by including the received session identifier in an HTTP Authorization header. Upon receiving such a request, the PEP extracts the session identifier and examines in its authorization table if its status is *Authorized* and whether is can be used with the requested household identifier. If all checks succeed, the PEP forwards the request to the interoperability middleware.

V. Implementation and evaluation

We have implemented the proposed cloud wallet and VC issuer as .net 6.0 web applications. Additionally, we have implemented a PEP as a Python 3 WSGI application.[4] As a client application we are using a mobile application implemented using React Native.

An interesting parameter that affects the performance of our system is the lifetime of session identifiers: short-lived session identifiers would result in frequent interactions with the wallet, which adds round-trip delays, on the other hand a long-lived identifier (e.g., equal to the lifetime of a VC) will allow a client application to make authorized API requests even if the corresponding VCs have been revoked. In our implementation we are using long-lived session identifiers, i.e., their lifetime equals to the lifetime of the shortest-lived VC included in a VP. In order to handle VC revocation, we have extended the authorization table to include the *credentialStatus* properties of the VCs that correspond to each session; then, a background process periodically fetches the revocation lists and updates the *status* of each identifier accordingly. It is reminded that a revocation list includes the status of all non expired VCs, therefore the background process has to perform as many requests as the number of issuers. Furthermore, our system achieves the following security properties:

a) Improved security management: User and access control policy management in our system is implemented independently of the interoperability middleware, since, granting or revoking an access right does not involve any communication with the middleware (as opposed for example to a system where Access Control Lists are stored in the middleware). Also, each issuer is allowed to use its internal policy for deciding who can access which household. Finally, from our solution perspective, introducing a new EMS component provider requires only updating the middleware with the new issuer URL and public key.

b) Attack surface reduction: In our solution the amount of verifications the PEP needs to perform is less compared to a system that relies on Access Control Lists (ACLs). Furthermore, the PEP is not required to store any additional secret information, nor does it have to maintain user accounts. Similarly, end-user client applications are not involved in the authorization process, since this is handled by the cloud wallet. Therefore, from a security perspective, creating a new

[4]URLs to the github repositories of the corresponding implementations will be provided in the camera ready version of the paper.

application for our system is easier and less prone to errors, compared to a solution where end-user client applications need to manage secrets.

c) Resilience to attacks: Since VCs in our system are bound to public key controller by the wallet, our system is not affected by attackers-in-the-middle that intercept the communication between a client application and the interoperability middleware. These attackers, can neither modify the transmitted VP without being detected, nor re-use the captured VP to their own purposes.

VI. Conclusions and future work

In this work, an access control solution using Verifiable Credentials (VCs) is introduced to facilitate trusted data access in the context of smart energy grids. Our approach is applied in systems that support multiple EMS component providers and integrate an interoperability middleware to facilitate the communication among them. Our access control mechanism allows the simultaneous usage of multiple authorizations, issued by different EMS component providers, hiding at the same time the authorization process from the middleware, by using a PEP acting as transparent proxy, as well as from end-user client applications, by using a cloud wallet. Furthermore, all process in our system leverage standards and ongoing specifications based on OpenID connect, therefore existing user management systems can be easily integrated.

Future work in this direction is focused on two areas. Firstly, our system can be expanded to enable finer-grained authorizations, e.g., to the device level. This feature could be used for example for authorizing a guest to access a portion of the devices included in a household. Secondly, we investigate the integration of our system with ongoing decentralized identification and authorization efforts which are based on related technologies, e.g., the new European digital identify architecture.[5]

Acknowledgements

This work was supported by project InterConnect (Interoperable Solutions Connecting Smart Homes, Buildings and Grids), which has received funding from European Union's Horizon 2020 Research and Innovation Programme under Grant 857237.

References

[1] S. Chadoulos, I. Koutsopoulos, and G. C. Polyzos, "Mobile apps meet the smart energy grid: A survey on consumer engagement and machine learning applications," *IEEE Access*, vol. 8, pp. 219 632–219 655, 2020.

[2] L. Daniele, F. den Hartog, and J. Roes, "Created in close interaction with the industry: the smart appliances reference (saref) ontology," in *Formal Ontologies Meet Industry: 7th International Workshop, FOMI 2015, Berlin, Germany, August 5, 2015, Proceedings 7*. Springer, 2015, pp. 100–112.

[3] J. M. Terras, T. Simão, D. Rua, F. Coelho, C. Gouveia, R. Bessa, J. Baumeister, R.-I. Prümm, O. Genest, A. Siarheyeva *et al.*, "Fostering the relation and the connectivity between smart homes and grids–InterConnect project," in *CIRED 2020 Berlin Workshop (CIRED 2020)*, vol. 2020. IET, 2020, pp. 761–764.

[4] Sporny, M. et al., "Verifiable credentials data model 1.1," W3C, W3C Recommendation, 2022, https://www.w3.org/TR/verifiable-claims-data-model/.

[5] ——, "Verifiable credentials status list v2021," W3C, W3C Draft Community Group Report, 2023, https://www.w3.org/TR/vc-status-list/.

[6] T. Lodderstedt, K. Yasuda, and T. Looker, "OpenID for Verifiable Credential Issuance," OpenID Connect WG, Internet draft, 2023, https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.

[7] O. Terbu, T. Lodderstedt, K. Yasuda, and T. Looker, "OpenID for Verifiable Presentations - draft 18," OpenID Connect WG, Internet draft, 2023, https://openid.net/specs/openid-4-verifiable-presentations-1_0.html.

[8] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Signature (JWS)," Internet Requests for Comments, IETF, RFC 7515, May 2015. [Online]. Available: https://tools.ietf.org/html/rfc7515

[9] L. Tan, N. Shi, C. Yang, and K. Yu, "A blockchain-based access control framework for cyber-physical-social system big data," *IEEE Access*, vol. 8, pp. 77 215–77 226, 2020.

[10] F. Chen, J. Huang, C. Wang, Y. Tang, C. Huang, D. Xie, T. Wang, and C. Zhao, "Data access control based on blockchain in medical cyber physical systems," *Security and Communication Networks*, vol. 2021, pp. 1–14, 2021.

[11] J. Koo, G. Kang, and Y.-G. Kim, "Interoperable access control framework for services demanding high level security among heterogeneous iot platforms," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '23. New York, NY, USA: Association for Computing Machinery, p. 737–740.

[12] A. Birgisson, J. G. Politz, Úlfar Erlingsson, A. Taly, M. Vrable, and M. Lentczner, "Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud," in *Network and Distributed System Security Symposium*, 2014.

[13] C. L. Webber, M. Sporny Eds, "Authorization capabilities for linked data v0.3," W3C, W3C Draft Community Group Report, 2023, https://w3c-ccg.github.io/zcap-spec/.

[14] M. P. Andersen, S. Kumar, M. AbdelBaky, G. Fierro, J. Kolb, H.-S. Kim, D. E. Culler, and R. A. Popa, "WAVE: A Decentralized Authorization Framework with Transitive Delegation," in *Proceedings of the 28th USENIX Conference on Security Symposium*, ser. SEC'19. USA: USENIX Association, 2019, p. 1375–1392.

[15] D. Longley and M. sporny, "Credential handler api 1.0," W3C, W3C Draft Community Group Report, 2021, https://w3c-ccg.github.io/credential-handler-api/.

[16] Curren, S. et al., Identity Foundation, DIF Ratified Specification, 2023, https://identity.foundation/didcomm-messaging/spec/.

[17] Q. Zhou, M. Elbadry, F. Ye, and Y. Yang, "Heracles: Scalable, fine-grained access control for internet-of-things in enterprise environments," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 1772–1780.

[18] A. S. Li, R. Safavi-Naini, and P. W. L. Fong, "A capability-based distributed authorization system to enforce context-aware permission sequences," ser. SACMAT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 195–206.

[19] R. Schuster, V. Shmatikov, and E. Tromer, "Situational access control in the Internet of Things," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 1056–1073. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243817

[20] N. Fotiou, V. A. Siris, and G. C. Polyzos, "Capability-based access control for multi-tenant systems using OAuth 2.0 and Verifiable Credentials," in *30th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021.

[21] N. Fotiou, V. A. Siris, G. C. Polyzos, Y. Kortesniemi, and D. Lagutin, "Capabilities-based access control for iot devices using verifiable credentials," in *2022 IEEE Security and Privacy Workshops (SPW)*, 2022, pp. 222–228.

[22] M. Tosic, F. A. Coelho, B. Nouwt, D. E. Rua, A. Tomcic, and S. Pesic, "Towards a cross-domain semantically interoperable ecosystem," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1640–1641.

[23] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," IETF, RFC 7519, 2015.

[5] https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-architecture-and-reference-framework-outline