Data integrity protection for data spaces

Nikos Fotiou[*], George Xylomenos[†], Yannis Thomas[†]
[*]Excid, Athens, Greece. Email: fotiou@excid.io
[†]Mobile Multimedia Laboratory, Athens University of Economics and Business, Greece.
Email: {xgeorge,thomasi}@aueb.gr

***Abstract*—Data spaces are an emerging concept that allows intermediaries to facilitate data exchange among interested stakeholders. Often, these intermediaries are trusted to filter the relayed data items, e.g., in order to support query-based data access APIs, or to implement access control. In this paper, we explore and compare two approaches for protecting data against illegitimate tampering, balancing the need for data filtering and data integrity protection. We apply our concept in data spaces that serialize data objects using JSON-LD, e.g., data spaces that implement ETSI's NGSI-LD API, and we enable intermediaries to hide segments of the transmitted data, providing at the same time integrity verification proofs for the revealed portions. Both approaches are efficient, with minimal communication and computational overhead.**

***Index Terms*—Zero-knowledge proofs, NGSI-LD, Selective disclosure**

I. Introduction

Data spaces are emerging as a new form of digital platform aiming to liberate data from silos in order to enable data-driven innovations and shape digital transformation [1]. A growing number of reports by commercial entities and governmental bodies highlight the business potential and the possible societal impact that can be achieved by embracing data spaces (see for example [2]). Furthermore, data spaces are an integral part of EU's Data Governance Act[1], which is the first major legislative initiative implementing the European Strategy for Data.

A data space is composed of building blocks that enable semantic interoperability of data, uniform data access methods, as well as increased sovereignty and trust. Data spaces allow data *suppliers* to share the data that they control with data *consumers*. Data exchange is usually implemented through a data *intermediary*, which is responsible for controlling data access and how data is used [3]. Such data intermediaries are often implemented by third parties which provide an API for storing and accessing data (see also Figure 1).

In many use cases, data intermediaries should be able to selectively disclose portions of the transmitted data, e.g., when a consumer is only *interested* in some *attributes* of a data item, or when a consumer is *authorized* to access only specific attributes. In those cases, data integrity cannot be checked by using the traditional digital signatures generated by the data suppliers, since signature verification would fail for the partial data objects. Many existing systems simply delegate signing rights to the intermediary (e.g., in the form of a certificate chain). However, this enables intermediaries to modify the transmitted data, therefore it cannot be used when the intermediary is not *fully* trusted by the supplier.

[1]https://digital-strategy.ec.europa.eu/en/policies/data-governance-act

In this paper, we focus on data spaces where data is serialized using JSON-LD [4], a popular serialization format that combines JSON and data semantic interoperability. We propose two signing algorithms that achieve the following:

- **Secure Decomposition**: Data suppliers can decompose a data object into smaller fragments in a deterministic and reversible way and then generate a single digital signature that covers all fragments.
- **Secure Selective Disclosure**: Any third party–including data intermediaries–with access to the public key of the data supplier, the digital signature, and the fragments can produce a *verifiable* subset of the fragments.
- **Secure Selective Composition**: Given a *verifiable* subset of the fragments and the public key of the data supplier, any third party can verify the integrity of the given fragments and construct with them a new *composite* object.

Our solution allows intermediaries to hide segments of the transmitted data providing at the same time integrity verification proofs for the revealed portions. This process does not require intervention from the data suppliers, neither requires from intermediaries to store any secrets. Furthermore, it does not require data consumers to establish a trust relationship with the intermediaries.

The remainder of this paper is organized as follows. In Section 2 we present the design of our solution, detailing the underlay architecture and the signing algorithms. In Section 3 we present the implementation of our solution and its evaluation. We discuss related work in Section 4 and we conclude our paper in Section 5.

II. Design

A. Underlay architecture

Our data space is based on an intermediary implementing ETSI's NGSI-LD API [5]. This API allows access to *attributes* of *entities* representing real world assets, also known as *digital twins*. The attributes of an entity are defined by the entity *type*. As a motivating example, consider the use case of an intermediary (also referred to as *context broker* in NGSI-LD) providing information about cars located in a city. In this system there can be entities of type *car* with attributes such as *model*, *colour*, *speed*, *location*, as well as auxiliary entity types, e.g., *road* with attributes *name*, *traffic*, *geopath*, etc.

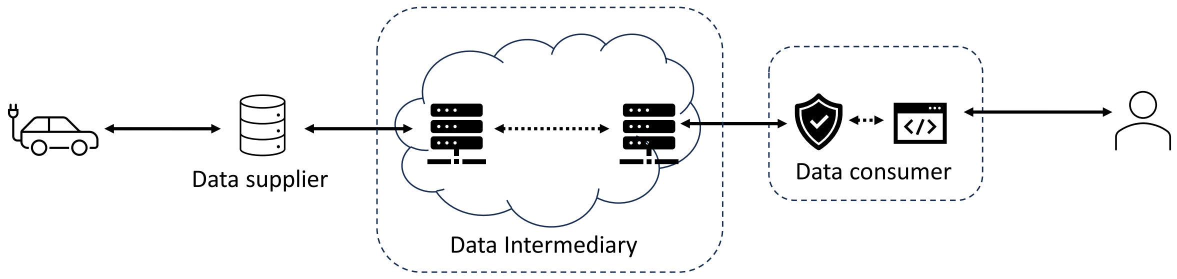


Fig. 1. The entities of our reference architecture.

An entity is represented as a JSON-LD object that includes[2] an *id* member with a URI as its value, a *type* member which contains the entity type, and a member for each entity attribute; the member's name is the attribute name and the member's value is the attribute value. An attribute value can be represented using any valid JSON data type, including another JSON object. Figure 2 shows on the left side a JSON-LD object used to represent an entity of type "car". As we can see, the value of member "colour" is a string, the value of member "speed" is an integer, whereas the value of member "brand" is another JSON object.

In the following subsection we present two designs that enable the selective disclosure of data. Both designs share a common step: the decomposition of a JSON-LD object into a list of *disclosures*. A disclosure represents a member of the JSON-LD object. For composite members, i.e., members whose value is a JSON object, a disclosure is constructed for all sub-members, following a depth-first approach. A disclosure is composed of two parts: the disclosure name whose value is the JSON pointer [6] to the corresponding member, i.e., a string representing the "path" to that member in the JSON-LD object, and the disclosure value which contains the corresponding member value. In the example of Figure 2, the table in the middle includes all possible disclosures for the provided JSON-LD object. It can be observed that there is a disclosure for the composite "brand" member (disclosure 5) and a separate disclosure for each of the members of the "brand" value (disclosures 6 and 7).

The decomposition process is reversible, i.e., given a set of disclosures, a JSON-LD object can be re-constructed in a straightforward way. Selective disclosure in our system is achieved by revealing only a portion of the disclosures, which are then combined by a data consumer into a *composite* JSON-LD object. In the example of Figure 2, using only the disclosures highlighted in the table (1, 4 and 6), we can construct the composite object shown on the right side.

The integrity of the revealed disclosures can be verified either using a digital signature over the hashes of all disclosures, or using BBS+ signatures. We consider both approaches in the following section.

B. Hash signature-based approach

In our first design, which is based on legacy and widely used cryptographic primitives, an object is signed as follows:

[2]In this paper we adopt the "concise" representation specified in section 4.5.2.3 of [5]

1) Initially, a data supplier calculates the disclosures of a JSON-LD object and transforms each disclosure into a single message by concatenating the disclosure name with the disclosure value, plus a random *salt* value, separating them using the space character.
2) For each message constructed in step 1, the data supplier calculates its hash.
3) The data supplier concatenates the base64 encoding of all hashes into a data structure and digitally signs it: the output of this process (the hashes and their digital signature) is used as the signature of the data object.
4) The data supplier stores the signature of the object, the disclosures and the corresponding salt values in the intermediary.

An intermediary can now reveal to a consumer a portion of the disclosures. A consumer can verify their integrity using the signature of the data object and construct a composite object, as follows:

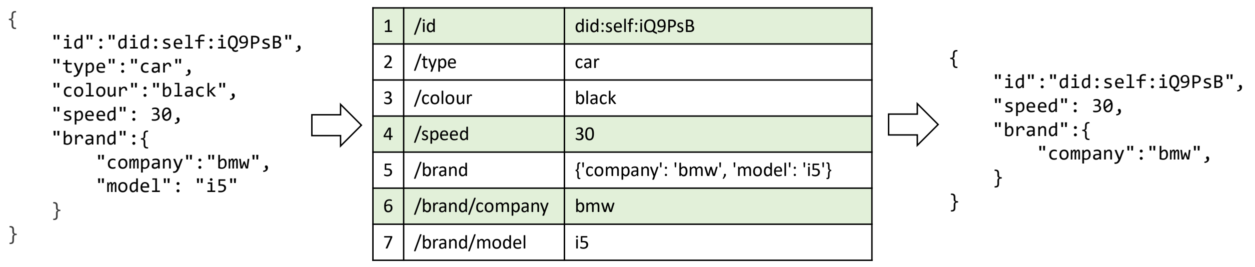
1) The consumer validates the signature of the data object using the public key of the data supplier.
2) For the available disclosures, the consumer reconstructs the messages that the supplier created in the first step of the signing algorithm (including the salt values).
3) For each message, the consumer calculates its hash and verifies that it is included in the signature of the data object.
4) Finally, the consumer creates a composite JSON-LD file using the name and value of each provided disclosure.

It can be easily observed that the consumer does not learn any information about the hidden disclosures, since the signature of the data object includes only their (salted) digest.

C. BBS+ signature-based approach

The BBS+ signature design is based on the group signature scheme presented in [7] enhanced with Zero-Knowledge Proofs (ZKPs) by [8] and [9]. The BBS+ signature scheme is currently under standardization by CFRG IETF group [10]. This scheme can be thought as a composition of two (interdependent) pairs of algorithms; one pair for generating and verifying group signatures, and another for generating and verifying ZKPs.

The BBS+ signature algorithm enables digital signatures over a group of individual messages. This algorithm accepts as input the group of messages and the signer's private key, and outputs a single *constant size* signature. The signature



```
{
    "id":"did:self:iQ9PsB",
    "type":"car",
    "colour":"black",
    "speed": 30,
    "brand":{
        "company":"bmw",
        "model": "i5"
    }
}
```

1	/id	did:self:iQ9PsB
2	/type	car
3	/colour	black
4	/speed	30
5	/brand	{'company': 'bmw', 'model': 'i5'}
6	/brand/company	bmw
7	/brand/model	i5

```
{
    "id":"did:self:iQ9PsB",
    "speed": 30,
    "brand":{
        "company":"bmw",
    }
}
```

Fig. 2. A JSON-LD object, its disclosures, and a composite object constructed using the 1st, 4th, and 6th disclosures.

can be validated given the signer's *Public Key* (PK) and the entire group of signed messages; this is equivalent to validating a "traditional" digital signature, if we consider the group of messages as a single compound message.

The BBS+ ZKP generation protocol is a non-interactive ZKP protocol that enables any entity that knows a signature generated using the signature algorithm and the original signed group of messages, to create a proof of knowledge of the signature while selectively disclosing only a sub-group of the signed messages. The size of the proof is linear to the number of hidden messages. The proof can be validated with only the signer's PK and the sub-group of revealed messages. The whole protocol is zero-knowledge in the sense that, from this process, no information can be derived about either the signature or the hidden messages.

In our design, an object is signed using BBS+ as follows:

1) Initially, a data supplier calculates the disclosures of a JSON-LD object and transforms each disclosure into a single message by concatenating the disclosure name and value, separating them using the space character.
2) Then, it generates a BBS+ signature providing as input its private key and the list of disclosures.
3) Finally, the JSON-LD object and the signature are stored in a intermediary.

An intermediary can reveal to a consumer a portion of the disclosures by calculating the corresponding ZKP. A consumer can verify their integrity using the provided ZKP and construct a composite object as follows:

1) For the available disclosures, the consumer reconstructs the messages that the data supplier calculated in the first step of the signing algorithm.
2) The consumer verifies that the provided ZKP is a valid proof for the calculated messages.
3) Finally, the consumer extracts the name and value of each disclosure and creates a composite JSON-LD object.

Similarly to the hash-based approach, the ZKP approach does not reveal any information about the hidden disclosures.

III. Implementation and evaluation

We have implemented a data space for sharing data generated by entities representing (emulated) cars (see also Figure 1). Our intermediary is implemented as a distributed brokering system.[3] A data supplier stores data disclosures and the corresponding signatures in the intermediary. An intermediary is accessed through an NGSI-LD API gateway, which is trusted by the data consumer. This gateway, which may be administered by the data consumer, is implemented using the Python Web-Server Gateway Interface.[4] The gateway is responsible for receiving NGSI-LD API requests from a data consumer client application, retrieving the corresponding disclosures from the intermediary, validating the proofs, and responding with the composite object. We consider API requests where the consumer is interested in READing only certain attributes of a data object, therefore, the brokering system provides only the corresponding disclosures.

A. Performance evaluation

1) Set up: We start with a baseline scenario where the brokering system is fully trusted, hence it constructs the composite object, encloses it in a JSON Web Signature (JWS) [11] using ECDSA with the P-256 curve, and transmits it to the NGSI-LD API gateway. We construct artificial JSON-LD objects consisting of 100 members. Member names and values are randomly generated 5-character strings.[5]

For the hash-based option, salts are 128-bit random numbers encoded using base64, hashes of disclosures are calculated using SHA-256, encoded using base64 and stored as a JSON array; then this array is enclosed in a JWS using ECDSA with the P-256 curve: this is the signature of the data object. Released disclosures are transmitted as a base64-encoded JSON array, followed by the generated signature of the data object.

For the BBS+ signature option, we rely on MATTR's BBS+ implementation[6] which generates group signatures using the BLS12-381 pairing-friendly elliptic curve [12]. Released disclosures are transmitted as a base64-encoded JSON array, followed by the base64 encoded ZKP.

2) Communication overhead: We measure the communication overhead between the brokering system and the NGSI-LD API gateway as a function of the number of revealed disclosures. In particular we measure the size of a response sent from the brokering system and the NGSI-LD API gateway when 1-100 disclosures are revealed. Figure 3 shows that the hash-based approach has the highest communication overhead. We can also see that as the number of revealed disclosures

[3]More information about the implemented system can be found at https://mmlab-aueb.github.io/snds-site/

[4]https://wsgi.readthedocs.io/

[5]Our implementation and the evaluation scenarios can be found at https://github.com/mmlab-aueb/selective-disclosure

[6]https://github.com/mattrglobal/ffi-bbs-signatures

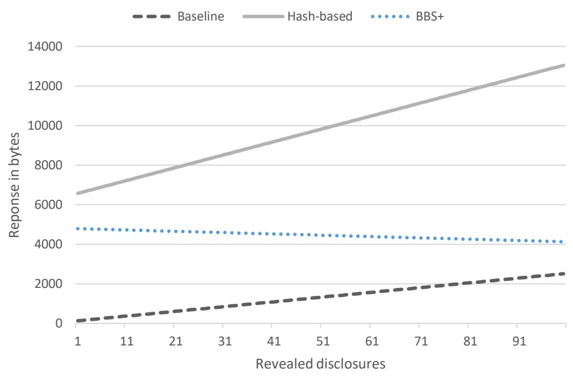


Fig. 3. Communication overhead.

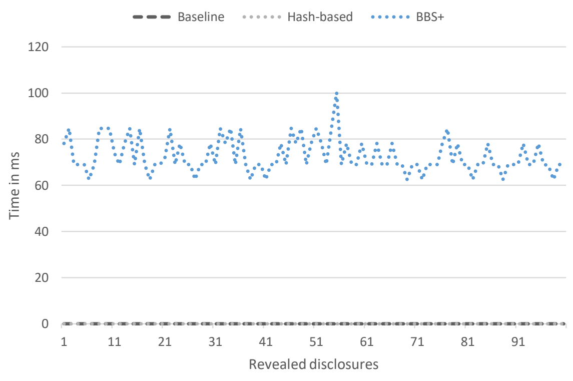


Fig. 4. Computational overhead of the intermediary.

increases, the communication overhead of the BBS+ approach decreases, since the more disclosures are revealed, the smaller the size of the ZKP is.

3) Computational overhead: We measure the time required for a supplier to calculate a signature, for the intermediary to reveal some disclosures, and for the consumer (i.e., the NGSI-LD API gateway) to verify the integrity of the received disclosures. These measurements are obtained in a desktop PC using an Intel i5 processor and 8GB of RAM, running Ubuntu 22.04.

In the baseline scenario, the supplier does not calculate any signatures, whereas in both our designs the signature calculation time is constant. Specifically, in the hash-based approach the time required for calculating the signature of the object is $< 0.1ms$, whereas in the BBS+ approach the time required for calculating the group signature is $\approx 22ms$.

The computational overhead of an intermediary may depend on the number of revealed disclosures. As shown in Figure 4, the computational overhead of an intermediary in the hash-based approach is 0, since the intermediary does not have to perform any cryptographic operations; similarly, in the baseline scenario an intermediary calculates a JWS that requires $< 0.1ms$. In contrast, in the BBS+ approach, an intermediary needs on average $70ms$ to produce a response, as it needs to calculate a ZKP.

Similarly, the computational overhead of a consumer may

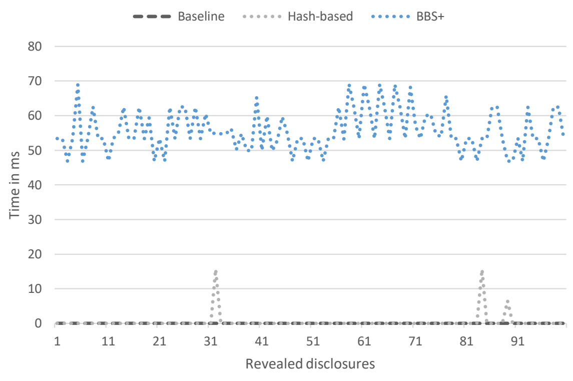


Fig. 5. Computational overhead of the consumer.

also depend on the number of revealed disclosures. As shown in Figure 5, the computational overhead of a consumer in the hash-based approach is $< 0.1ms$ since the consumer only has to verify a signature, calculate some hash functions and perform a lookup; similarly, in the baseline scenario a consumer only has to validate a signature, which requires $< 0.1ms$. In the BBS+ approach, a consumer needs on average $60ms$ to verify a response, as it needs to validate a ZKP.

B. Security evaluation

Our threat model considers attackers acting as intermediaries, wishing to modify relayed data, as well as honest but curious gateways wishing to learn information about the hidden disclosures.

It can be trivially proved that our solution protects the integrity of the revealed disclosures. Indeed, in the hash-based approach, an invalid disclosure would be accepted by a conforming consumer only if its hash was included in the signature of the data object (created in step 3 of the signing algorithm). However, this would require from an attacker either to find a collision in the used hash function, so that the hash of the invalid disclosure would match the hash of a valid disclosure, or to break the security of JWS, allowing the hash of the invalid disclosure to be substituted in the signature of the data object. Similarly, in the BBS+ approach, an invalid disclosure would be accepted by a conforming consumer only if an attacker could break the security of the BBS+ signature algorithm.

Furthermore, both designs considered by our solution achieve *indistinguishability* of hidden disclosures. Specifically, let two disclosures, D_1 and D_2 that belong to different objects but have the same name and value (e.g., two objects of type “car” of the same “colour”); if D_1 and/or D_2 are hidden, a curious gateway cannot tell if the relationship $D_1 == D_2$ holds. In the hash-based approach this is achieved by randomizing the output of the hash function used for creating the signature of the data object by introducing some random salt. Similarly, in the BBS+ approach, all ZKPs are randomized.

On the other hand, *untraceability* of disclosures is only achieved by the BBS+ based approach. Specifically, let D_{t1} be the disclosure of an object at $time1$ and D_{t2} the same disclosure of the same object at $time2$ (e.g., the speed of a car at $time1$ and the speed of the same car at $time2$). Even if D_{t1} and/or D_{t2} are hidden, a a curious gateway can tell whether or not $D_{t1} == D_{t2}$ in the hash-based approach. This is not the case in the BBS+ based approach, since the output of the ZKP process is randomized.

Finally, although both designs do not provide any information about the name and the value of the hidden disclosures, both reveal the number of the hidden disclosures. In case this information is sensitive, decoy disclosures can be used.

IV. Related work

A simple approach for providing functionality similar to our system is the decomposition of a JSON-LD object into smaller "objects" with each such object being individually signed. Such an approach has been followed in similar contexts by related work (e.g., [13]). Nevertheless, this approach has some disadvantages compared to our solution: it results in bigger messages, it requires more signature verifications by the data consumers, and it requires auxiliary information stored in each "smaller object" to indicate that they are indeed fragments of the original object.

Our hash-based approach is related to *Content Extraction Signatures* (CES) [14], [15], [16]. Our solution is specific to JSON-LD objects and it provides constructs that can be used for decomposing a JSON-LD object into multiple "messages", a process also required by CES but not specified elsewhere. Similarly, some CES realizations organize hashes of messages into a Merkle tree and sign the root of the tree (as opposed to the whole list of hashes). This can also be used in our system.

Similarly, our hash-based approach shares similarities with the Selective Disclosure for JWTs (SD-JWT) [17] IETF draft. This draft focus on JSON Web Tokens (JWT) and uses disclosures to selectively reveal members of a JWT. Moreover, this draft also stores salted hashes of the disclosures in the JWT, which is signed by the corresponding "issuer". The main difference between this draft and our solution is that an SD-JWT does not use JSON Pointers to specify the disclosure name, instead using the corresponding attribute name as the disclosure name and defines inside the JWT "placeholders" where a disclosure should be put to construct a composite object. Our solution does not use such placeholders, instead it allows composite object creation using only the disclosures. Using this approach, and as opposed to SD-JWTs, our solution makes it easier to "hide" members of composite objects, or even array elements. Furthermore, our approach can be used with BBS+ signatures in a straightforward manner.

Many systems implement selective disclosure via *Attribute-Based Encryption* (ABE) (see for example [18]). ABE allows data suppliers to encrypt their data in such a way that only consumers that have specific "attributes" can decrypt it. These systems, unlike our solution, also protect data confidentiality against intermediaries. Therefore, in those systems, data intermediaries are mere brokers of encrypted items. In our solution data intermediaries have access to the contents of a data item hence they can provide more advanced functionality, e.g., they can support API calls that include data filtering criteria. Furthermore, in encryption-based systems all data items usually have to be re-encrypted every time a key is breached. Other recent systems use Homomorphic Encryption to allow intermediaries to perform some operations (see for example [19]). However, the overhead of homomorphic encryption is not tolerable for many use cases.

V. Conclusions

In this paper we considered the problem of data integrity protection in data spaces that include third party intermediaries. We presented the design, implementation, and evaluation of two approaches that enable intermediaries to hide segments of the relayed data objects, providing at the same time integrity protection for the revealed portion of the data objects. Both approaches do not require any intervention by the data supplier, nor do they require the intermediaries to store any secret information. Furthermore, both approaches introduce low computational and communication overhead.

Both solutions enable a data intermediary to hide any members of the data objects stored in the data space. Future work in this area includes tools for allowing data publishers to specify members that cannot be hidden (e.g., the object identifier). Furthermore, our implementation relied on a custom-implemented NGSI-LD API gateway. Future work in this area includes the integration of our solution to existing systems, such as the FIWARE Orion context broker.[7] Finally, in our solution composite objects were constructed by the NGSI-LD gateway; we are investigating protocols that allow the integration of our proofs directly into NGSI-LD objects (such as the Data Integrity proofs under standardization by W3C): this will not only allow intermediaries to construct the composite objects by themselves, but it will also enable client applications to verify the integrity of the received objects, following standards-compliant approaches, and without relying on a trusted gateway.

Acknowledgement

The work reported in this paper has been partly funded by the EU's Horizon 2020 Programme through the subgrant Secure Named Data Sharing (SNDS) (NGISARGASSO-2023-CALL1-9-SNDS) of project NGI SARGASSO (grant agreement No 101092887).

References

[1] D. Beverungen, T. Hess, A. Köster, and C. Lehrer, "From private digital platforms to public data spaces: implications for the digital transformation," *Electronic Markets*, vol. 32, no. 2, pp. 493–501, 2022.

[2] S. Scerri, T. Tuikka, I. L. de Vallejo, and E. Curry, "Common european data spaces: Challenges and opportunities," *Data Spaces: Design, Deployment and Future Directions*, pp. 337–357, 2022.

[3] M. M, F. E, C. S. B, P. S. M, S. S, and V. M, "Mapping the landscape of data intermediaries," EU Publications Office, Scientific analysis or review, 2023.

[4] Manu Sporny et al., "JSON-LD 1.1, A JSON-based Serialization for Linked Data," W3C Recommendation, 2020.

[5] Context Information Management (CIM) ETSI ISG, "NGSI-LD API," ETSI, Group Specification CIM-009v161, 2022.

[6] P. Bryan (ed.), "Javascript object notation (json) pointer," IETF, RFC 6901, 2013.

[7] https://fiware-orion.readthedocs.io/en/master/

[7] D. Boneh, X. Boyen, and H. Shacham, “Short group signatures,” in *Annual International Cryptology Conference*. Heidelberg, DE: Springer, 2004, pp. 41–55.

[8] M. H. Au, W. Susilo, and Y. Mu, “Constant-size dynamic k-taa,” in *International Conference on Security and Cryptography for Networks*. Heidelberg, DE: Springer, 2006, pp. 111–125.

[9] J. Camenisch, M. Drijvers, and A. Lehmann, “Anonymous attestation using the strong diffie hellman assumption revisited,” in *International Conference on Trust and Trustworthy Computing*. Heidelberg, DE: Springer, 2016, pp. 1–20.

[10] T. Looker, V. Kalos, A. Whitehead, and M. Lodder, “The bbs signature scheme,” IETF, Tech. Rep., 2023, https:// www.ietf.org/archive/id/draft-irtf-cfrg-bbs-signatures-04.html.

[11] 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

[12] P. S. L. M. Barreto, B. Lynn, and M. Scott, “Constructing elliptic curves with prescribed embedding degrees,” pp. 257–267, 2003.

[13] G. Miklau and D. Suciu, “Managing integrity for data exchanged on the web,” in *WebDB*, 2005, pp. 13–18.

[14] R. Steinfeld, L. Bull, and Y. Zheng, “Content extraction signatures,” in *Proceedings of the 4th International Conference on Information Security and Cryptology (ICISC)*. Springer, 2002, pp. 285–304.

[15] L. Bull, P. Stanski, and D. M. Squire, “Content extraction signatures using xml digital signatures and custom transforms on-demand,” in *Proceedings of the 12th International Conference on World Wide Web (WWW)*. New York, NY, USA: ACM, 2003, p. 170–177.

[16] L. Bull, D. M. Squire, J. Newmarch, and Y. Zheng, “Grouping verifiable content for selective disclosure,” in *Information Security and Privacy*, R. Safavi-Naini and J. Seberry, Eds. Heidelberg, DE: Springer, 2003, pp. 1–12.

[17] D. Fett, K. Yasuda, and B. Campbell, “Selective disclosure for jwts (sd-jwt),” IETF, Tech. Rep., 2023, https:// datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/.

[18] J. L. Hernández-Ramos, S. Pérez, C. Hennebert, J. B. Bernabé, B. Denis, A. Macabies, and A. F. Skarmeta, “Protecting personal data in iot platform scenarios through encryption-based selective disclosure,” *Computer Communications*, vol. 130, pp. 20–37, 2018.

[19] Y. Hong, L. Yang, Z. Xiong, S. S. Kanhere, and H. Jiang, “Ochjrn-chain: A blockchain-based security data sharing framework for online car-hailing journey,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2023.