Authentication, Authorization, and Selective Disclosure for IoT data sharing using Verifiable Credentials and Zero-Knowledge Proofs

Nikos Fotiou[1], Iakovos Pittaras[1], Spiros Chadoulos[1,2], Vasilios A. Siris[1], George C. Polyzos[1], Nikolaos Ipiotis[2], and Stratos Keranidis[3]

[1] Mobile Multimedia Laboratory, Department of Informatics
School of Information Sciences and Technology
Athens University of Economics and Business
Evelpidon 47A, 113 62 Athens, Greece
{fotiou,pittaras,spiroscha,vsiris,polyzos}@aueb.gr
[2] Plegma Labs
Neratziotissis 115, 15124, Marousi, Athens, Greece
{sc,ni}@pleg.ma
[3] DomX IoT Technologies
Stratigou Sarafi 48E, 55133, Thessaloniki, Greece
stratos@domx.io

Abstract. As IoT becomes omnipresent vast amounts of data are generated, which can be used for building innovative applications. However, interoperability issues and security concerns, prevent harvesting the full potentials of these data. In this paper we consider the use case of data generated by smart buildings. Buildings are becoming ever "smarter" by integrating IoT devices that improve comfort through sensing and automation. However, these devices and their data are usually siloed in specific applications or manufacturers, even though they can be valuable for various interested stakeholders who provide different types of "over the top" services, e.g., energy management. Most data sharing techniques follow an "all or nothing" approach, creating significant security and privacy threats, when even partially revealed, privacy-preserving, data subsets can fuel innovative applications. With these in mind we develop a platform that enables controlled, privacy-preserving sharing of data items. Our system innovates in two directions: Firstly, it provides a framework for allowing discovery and selective disclosure of IoT data without violating their integrity. Secondly, it provides a user-friendly, intuitive mechanisms allowing efficient, fine-grained access control over the shared data. Our solution leverages recent advances in the areas of Self-Sovereign Identities, Verifiable Credentials, and Zero-Knowledge Proofs, and it integrates them in a platform that combines the industry-standard authorization framework OAuth 2.0 and the Web of Things specifications.

1 Introduction

IoT systems generate vast amounts of data nevertheless, their potential is limited by security and privacy concerns, as well as by the lack of interoperability. A striking example is the case of smart buildings. Smart buildings employ a variety of IoT devices that generate data which support various applications, such as energy management, automations, security and safety, etc. These applications are in most cases siloed and the generated data are only used for the specific purposes of each application. Nevertheless, these data can be valuable for a variety of stakeholders that are able to deliver value-added services for other domains. Energy suppliers represent a key stakeholder that can significantly benefit from both energy and non-energy data that can be collected, either directly by smart building systems or even by legacy systems that are integrated with smart IoT equipment. According to an Accenture study [11], energy utilities will need to master data analytics in the near future, to continue developing valuable, customer-focused products that go far beyond old business models and plain commodity offerings. Data analytics can benefit energy utilities in multiple ways: a) successful retention of customers through the delivery of innovative personalized services, b) improved customer targeting and segmentation through consumer profiling, c) improved energy market participation through demand forecasting based on machine learning, d) improved energy savings for end users through optimized demand management and many others.
On the other hand, end-users would be interested in securely making a subset of the data generated by their IoT devices available to these 3rd parties, in a stratified manner, to benefit from the added value of the provided services. Nevertheless, several challenges have to be overcome: a) a uniform and standardized way for advertising/discovering, requesting, and transmitting data should be in place, b) sensitive information should be stripped from the shared data without violating data integrity and provenance, c) an efficient, usable mechanism for expressing and enforcing fine grained access control policies should be available, d) data access rights should be expressed in a rich and verifiable manner. In addition to overcoming these challenges, proposed solutions should encourage interoperability and prevent vendor "lock-in". With these in mind we designed, implemented, and evaluated *SelectShare*: a platform for controlled sharing of IoT data, focusing on smart buildings.
SelectShare is a system that makes available data from IoT systems located in buildings, and facilitates fine-grained, privacy-preserving data access to controlled subsets of these data, while at the same time ensuring data integrity, provenance verification, authenticity, and interoperability with different types of systems. This is achieved by integrating four components. First, an IoT gateway that exposes a data access API by following W3C's Web of Things specifications [13] facilitating data discovery and data interoperability. Second, a data transcoder that collects data from IoT devices, transcodes them into JSON objects, and signs them using a digital signature scheme that enables selective disclosure of the claims included in the JSON, providing at the same time Zero-Knowledge Proofs (ZKPs) of their integrity. Third an OAuth 2.0 [9] based Verifiable

Credential (VC) [15] issuing mechanism for generating self-contained, fine-grained access tokens. Finally, an HTTP-proxy that acts as a Policy Enforcement Point (PEP), for controlling access to the IoT gateway, as well as for selectively hiding parts of the responses generated by the gateway. Using this approach, SelectShare achieves fine-grained access control with minimal overhead and no modification to the IoT devices.
The remainder of this paper is structured as follows. In Section 2, we introduce our enabling mechanisms and we discuss related work in this area. In Section 3, we detail the design of our architecture. In Section 4, we present the implementation and evaluation of our solution. We conclude our paper in Section 5.

2 Background and related work

2.1 Verifiable Credentials

A *Verifiable Credential* (VC) [15] allows an *issuer* to assert some *attributes* about an entity referred to as the VC *subject*. A VC includes information about the issuer, the subject, the asserted attributes, as well as possible constraints (e.g., expiration date). Then, a VC *holder* (which is usually the same entity as the VC subject) can prove to a *verifier* that it owns a VC with certain characteristics. This is usually achieved by including in the VC an identifier (e.g., a public key), owned by the holder that enables the holder to generate a *proof of possession* (e.g., a digital signature with the corresponding private key). The VC verification process does not require communication with the VC 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. Our system uses a new VC type named *CapabilitiesCredential* that "describes" which portion of a data item a user can access.

2.2 BBS+ digital signatures

BBS+ is a digital signature protocol which is used for signing an *array* of messages. It was first envisioned by [2] (from where it takes its name), touched again in [1], re-visited in [3] and is currently under standardization [25]. BBS+ provide the ability to sign an array of individual messages, with only a single *constant size* signature. The signature can be validated given the signer's *Public Key* (PK) and the entire array of signed messages; this is equivalent to validating a "traditional" digital signature, if we consider the array of messages as a single compound message.
BBS+ can be combined with Zero-Knowledge Proofs (ZKP) allowing an entity to selectively hide elements of singed array of messages. In particular, *any* entity that knows the signature and the original signed array of messages, can create a proof of knowledge of the signature while selectively disclosing only a sub-array of the signed messages. The proof size will be linear to the number of un-revealed messages. The proof can be validated with only the signer's public key and the array of revealed messages.

2.3 Related Work

Related systems are using "attribute-based access control" (ABAC) (e.g., [8] [5]) for achieving similar goals. With ABAC, users own a "token" that includes their attributes. Then, a "policy decision point" (PDP) decides whether a user can perform a requested operation based on a list of pre-configured access control policies. Our system follows an alternative approach: our proposed solution is in essence a "capabilities-based access control" system where users own a token that describes their capabilities. The main advantage of this approach is that it removes the need for access control lists. On the other hand, we recognize that ABAC is useful when access control decision involves user context; in this case the policy decision process should receive as input attributes related to the context of the user. Our proxy can be easily extended to include related mechanisms (e.g., the system presented in [18]). Similarly our proxy can be extended to accommodate aspects such as user behavior (e.g., the solution presented in [7]).

Many systems leverage the blockchain technology to achieve similar targets (e.g., [20][19]). We postulate that blockchain overhead cannot be tolerated by a system like ours and a trusted proxy that would mediate the communication between the blockchain and our system would be required: this trusted proxy would negate any decentralization advantages of the blockchain technology. It should be highlighted that many VC systems rely on a blockchain to achieve their security properties. However, VCs in our solution do not need any blockchain-based system.

Kratos (initially described in [21] and then extended in [22]) is a system that wants to achieve similar goals as our solution for home IoT environments, where an IoT device may be owned by multiple users who may define different access control policies. Our solution considers that each IoT device is owned by a single entity, hence our approach is simpler. Additionally, Kratos proposes its own, specific mechanisms for expressing policies and rights, whereas our system relies on existing, open standards; hence our solution can be easily integrated in existing deployments.

Our solution assumes that IoT devices produce correct data and it does not consider any countermeasure against malicious IoT device owners. Our solution can be complemented by existing solutions that incentivize IoT device owner to provide correct data (e.g., [16]). Finally, in our solution, the used HTTP proxy is trusted to disclose the appropriate information; other related works rely on cryptographic constructions for not requiring this trust relationship (e.g., the work in [23] relies on "Attribute-based encryption"). However, this comes with the cost of additional computational overhead, as well as with the overhead of managing encryption keys.

Our solution extends our previous work presented in [6]. In SelectShare, we consider gateways that interconnect IoT devices that may be owned by different entities. Additionally, SelectShare assumes that the data generated by the IoT devices has been collected, singed, and stored in a storage node, prior being requested. Finally, SelectShare leverages ZKPs in order to provide even finer-grained access control.

3 Architecture

SelectShare architecture (also illustrated in Fig. 1) considers collections of IoT devices the belong to the corresponding IoT device *owner* (e.g., IoT devices of a smart building). These devices produce *measurements* that the device owners wish to share with 3^{rd} party data *clients* (e.g., analytics services). Data sharing is implemented through a single gateway, administrated by an independent service *provider* that can be accessed by clients using a standardized API. This gateway retrieves data from a *storage node*, which acts as a data repository, populated by specialized data *transcoders*. The communication between a client and the gateway is intercepted by a proxy which is responsible for validating client access rights, as well as for hiding parts of the response generated by the gateway. Clients' access rights are expressed using a Verifiable Credential (VC) issued by a VC issuer.

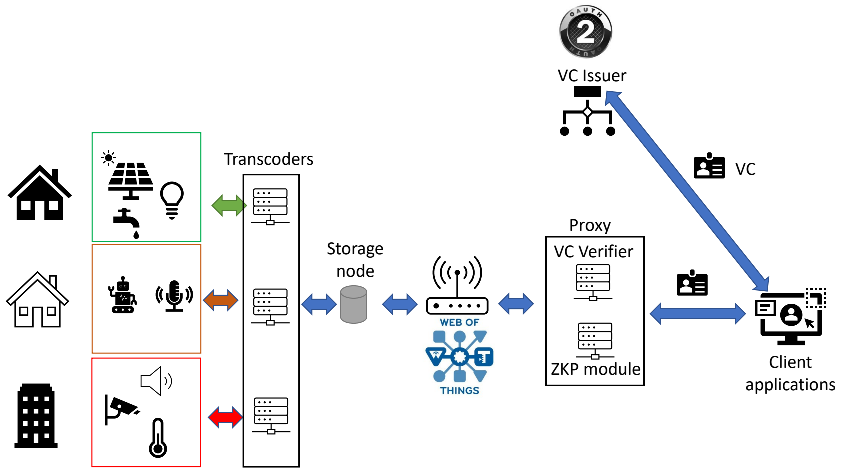

Fig. 1. Overview of the SelectShare architecture

SelectShare considers a setup phase during which: device owners configure VC issuers with the corresponding access control policies, and the proxy is configured with a list of trusted issuers per IoT device owner.

3.1 Data encoding and signing

In order to facilitate data sharing, SelectShare architecture considers an entity, named *transcoder*, which transcodes the data produced by each IoT device based on a predefined JSON *schema*. In our particular instantiation a generated JSON file includes: i) an IoT device specific identifier and ii) a list of measurements, where each measurement includes a device-unique measurement identifier (called *field*) and a list of *value-time* pairs. The following listing is an example of a generated JSON file.

```
{
  “deviceID”:“monitor−1”,
  “measurements”:[
      {
      “field”:“temperature”,
      “values”:[
          “time”:“1658162155”,
          “value”:“30C”
          ]
      },
      {
      “field”:“humidity”,
      “values”:[
          “time”:“1658162155”,
          “value”:“50”
        ]
      },
    ]
}
```

Listing 1.1. A JSON file produced by a transcoder

It should be highlighted that depending on the requirements of a SelectShare deployment, different schemas can be considered. A transcoder is owned and managed by the corresponding IoT devices owner, i.e., a transcoder interacts with the IoT devices of a specific owner. Additionally, each transcoder is configured with a BBS+ signing key and each generated JSON file is singed using BBS+ (by the transcoder). Finally, singed JSON files are stored in a storage node, administrated by the service provider.
Specific fields of a JSON file can be accessed over HTTP, through SelectShare's gateway, which implements Web of Things (WoT) Things Description (TD) [17] specifications. The WoT architecture attempts to structure well-known web protocols and tools for connecting IoT devices to the Web. In the WoT architecture communication model, IoT devices (real ones or virtual) are made available through REST-based APIs. To improve the interoperability and usability of IoT platforms, the WoT model uses a common format for describing IoT devices referred to as the Thing Description (TD). TD is a JSON-LD encoded file that includes metadata information about the IoT device (such as its id, a title, security definitions, etc), and defines API endpoints that can be used for accessing/invoking a device's properties, actions, and events.

3.2 Authentication and Authorization request

The VC issuer is an OAuth 2.0 authorization server extended with VC issuing capabilities. Issued VCs are encoded as JSON Web Tokens (JWT) and signed using JSON Web Signatures (JWS) (based on section 6.3 of [15]), improving compatibility and integration with existing tools. SelectShare considers VCs that describe which “measurements” of the IoT devices of a particular owner, a client can access. These VCs are generated based on policies defined by the corresponding IoT device owner. Additionally, a SelectShare VC issuer maintains a VC revocation list by implementing [14]. In particular, an issuer maintains a revocation list that concerns all non-expired VCs it has issued. This list is a simple bit-string and each VC is associated with a position in the list. Revoking a VC means setting the value of the bit corresponding to the VC equal to 1. Furthermore, each generated VC includes a field named "revocationListIndex" that specifies the position of the credential in the revocation list. Finally, a VC issuer is configured with client *credentials* (a client identifier and a client secret in our implementation), as well as with access control policies that map a client identifier to the corresponding access rights.

A client requests from the issuer a VC. A VC request is in essence an OAuth 2.0 access token request using the client credentials grant (section 4.4 of [9]), (in our implementation the corresponding client identifier and secret are used as the "credentials grant"). Additionally, the client generates a public-private key pair and instructs the issuer to include the generated public key in the issued VC. This is achieved using OAuth 2.0 Rich Authorization Requests [24]. In particular, the corresponding OAuth 2.0 access token request, is extended to include the generated public key (encoded as a JSON Web Key (JWK) [10]) and a digital signature generated using the corresponding private key. The issuer authenticates the client based on the included grant and generates a VC.

A VC is the base64url encoding of a JWT singed by the issuer, according to the VC data model. The generated JWT includes a *cnf* field, as specified by RFC 7800, that contains the public key generated by the client and included in the corresponding request. The VCs used in SelectShare are of type "CapabilitiesCredential". This type includes an array, called "capabilities", and each element of this array is a map that maps an IoT device identifier to a list of measurements the client can access. An example of a VC before encoding follows (the signature part is omitted).

```
{
    "jti": "https://issuer.com/credentials/1",
    "iss": "https://issuer.com",
    "aud": "owner−1"
    "iat": 1617559370,
    "exp": 1618423370,
    "cnf": {
      "jwk": <client jwk>
     },
    "vc": {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://mm.aueb.gr/contexts/capabilities/v1",
      ],
     "type": ["VerifiableCredential"],
      "credentialSubject": {
        "type": ["CapabilitiesCredential"],
                "capabilities": {
                    "monitor−1": [
                        "temperature",
                    ]
                }
            }
        }
    }
```

Listing 1.2. An example of a VC in our system

As it can be observed, a VC includes an identifier (the *jti* field), an identifier for the issuer (the *iss* field), an identifier for the IoT device owner (the *aud* field), an issuance and expiration time, the client public key, and the client's "capabilities". In the VC included in this example, a client can access the "temperature" measurement of "monitor-1" IoT device, owned by "owner-1".

3.3 Data access request

A client application requests to access some measurements of an IoT device by sending an appropriate HTTP request. This request includes the device identifier as a query parameter and a list of requested "fields" in a HTTP POST body. The HTTP request includes two HTTP headers: one that contains the JWT-encoded VC, and another that contains *a proof-of-possession* of the public key included in the VC; the latter proof is generated using OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP) [4]. A DPoP proof is essence a JSON Web Signature (JWS) that can be verified using the public key included in the corresponding VC. The payload of the JWS is constructed using a random nonce, the HTTP request method, the HTTP request URI, and a timestamp indicating the proof's creation time.

A data access request is intercepted by SelectShare's HTTP proxy. SelectShare's HTTP proxy includes a *VC verifier*: the VC verifier examines if the request includes an appropriate VC and then it verifies the validity, the status, and the ownership of a VC. A VC is appropriate if the "aud" claim includes the identifier of the device owner and if contains the "fields" of the "deviceID" included in the request.

The validity of a VC is verified by evaluating whether: a) the VC has not expired, b) the signature of the VC is valid, c) the VC has been issued by an issuer trusted by the device owner.

The status of the VC is verified by communicating with the VC issuer, and using the validation process described in [14]. I.e., in a nutshell, the verifier retrieves the revocation list (which is a bitstring), locates the bit that corresponds to evaluated VC, and examines the value of that bit.

Finally, the ownership of a VC is validated using the DPoP proof, i.e., the verifier verifies that the proof is adequately fresh, it includes a nonce not seen before, it includes the correct HTTP method and URI, and its signature can be verified using the public key included in the VC.

3.4 Data access response

Upon receiving an authorized request, the proxy forwards to the gateway. The gateway retrieves from a storage node a JSON file that includes *among other things* the requested fields, and forwards to the proxy. Finally, the proxy applies the *selective disclosure* process, in order to hide the fields not included in the client request. The selective disclosure process involves two algorithms: *framing* and *canonicalization*.

Framing Framing refers to the derivation of a "sub-item" from an item, that contains only part of the original one. Data framing is used to enable selective disclosure of the data item's information. More specifically, the framing algorithm accepts the original item and a frame as input and returns a new item that only contains the key-value pairs specified by the frame. The frame itself is a JSON structure that specifies the parts of the original item that should appear in the resulting one (and be disclosed in the end). For this purpose, the frame contains the keys that lead to the values that the prover will want to reveal. The framing algorithm used in SelectShare also includes special symbols that can be used for selecting specific elements in an array. For example, considering Listing 1.1 the following frame will reveal "the value of all measurements that include the field *temperature*":

```
  {
    "measurements": {
        "*":{
            "field":"temperature'',
            "values":{
                "*":{
                    "value":""
                }
            }
        }
    }
}
```

Listing 1.3. An example of frame used in SelectShare

Applying this frame in Listing 1.1 will result in the following object:

```
  {
    "measurements":[
        {
        "field":"temperature",
        "values":[
            "value":"30C"
            ]
        }
      ]
  }
```

Listing 1.4. Output of framing operation

Canonicalization As discussed previously, BBS+ signatures act on arrays of messages and not on structured data formats like JSON. In order for a transcoder to be able to sign a data item, as well as in order for the proxy to be able to derive ZKPs, data items must be canonicalized. Various canonicalization algorithms have been proposed by related efforts. A canonicalization algorithm serializes a JSON-encoded item into an array of messages, which can then be signed by a multi-message digital signature system like BBS+. There are various security requirements that those algorithms must be conformant with, in order to not compromise the security of the system. In this work, we are using the JCan algorithm [12] which is a lightweight, provably secure, JSON canonicalization proposal, designed to work with any data model.

Selective disclosure Any entity can generate a sub-item of a content item based on a frame and provide a ZKP that proves its correctness as follows. Initially, that entity applies the framing algorithm to derive the sub-item. After framing, the same entity canonicalizes the resulting sub-item, gets the array of messages that correspond to the revealed information (from the security properties of the canonicalization algorithm, this array is guaranteed to be a subset of the signed array that resulted from the canonicalization of the original item) and uses that array to derive a ZKP using BBS+.
The function of selective disclosure is implemented in a distributed manner by the transcoder and the ZKP module of the proxy. In particular, transcoders are responsible for signing the generated JSON objects using BBS+ signatures. The signed object is forwarded through the WoT gateway to the proxy. Then the ZKP module of the proxy is responsible for framing the signed object and for generating the corresponding ZKP. The framing operation is implemented by taking into consideration the requested “fields” option included. It should be highlighted that the proxy assumes that the user is authorized to access this field: this is true since if the user was not authorized, the incoming request would have been blocked during the VC verification process.

4 Implementation and evaluation

We have implemented SelectShare's issuer[4] as .net core web application. Similarly we have implemented SelectShare's HTTP proxy as a Python 3 application[5]. Finally, we implemented SelectShare's gateway based on Eclipse's Thingweb WoT gateway[6].
SelectShare introduces minimal communication overhead. VCs can be long-term (since they are bound to a public key owned by the client), hence client authorization does not have to take place often. Similarly, by using DPoP, a client can prove possession of its VC in a single message, i.e., there is no need for additional roundtrips. Moreover, the size of a

[4] https://github.com/mmlab-aueb/vc-issuer
[5] https://github.com/mmlab-aueb/py-verifier
[6] https://projects.eclipse.org/projects/iot.thingweb

VC and the corresponding proof is only few bytes. Finally, when it comes to VC status verification, a VC verifier can retrieve the revocation list once and use it for multiple requests. It is reminded that a revocation list is a bitstring that includes the status of non-expired VCs: since each VC corresponds to single bit, a revocation list may include thousands of VCs.

Similarly SelectShare introduces minimal computational overhead. VC verification process involves only the validation of two digital signatures as well as a lookup in a JSON object. Both operations are lightweight. When it comes to the overhead introduced to a proxy by the selective disclosure process we performed the following experiment in an Ubuntu 18.04 machine equipped with an intel i7-3770 CPU, 3.40GHz and 16GB of RAM. We constructed JSON measurement file composed of 100 fields each of which includes a single value. We calculate the time required to sign and verify sub-items that include form 1 to 99 values. Fig. 2 show the signature and verification time, measured in ms. It can be observed that as the number of items included in the sub-item increases, the signature creation time decreases. This happens because for each hidden item a proxy has to perform a number of multiplications. On the other hand, the signature verification time remains almost stable. It should be noted that these measurements are obtained without any "pre-calculation", however, in a real deployment a proxy can pre-calculate many of the computations required to create a ZKP.

4.1 Security properties

SelectShare considers the following trust relationships. An IoT device owner trusts: the VC issuer to issue an appropriate VC and correctly maintain the revocation list, the VC verifier module of the proxy to validate VCs and a proofs correctly, and the ZKP module of the proxy to not reveal "extra fields". A client trusts: the VC issuer to correctly maintain the revocation list, and the proxy to not perform "denial of service".

SelectShare facilitates security management and decreases attacks' surface. In particular, in SelectShare all access control policies are managed in a single point: the VC issuer. Adding, updating, or removing an access control policy involves no communication with the verifiers or the gateway. This is achieved by adopting the "capabilities-based access control" paradigm that removes the need for maintaining access control lists (as opposed for example to Role/Attribute-based access control). Similarly, the access control decision process is simple and the most "advanced" and error prone task is examining if the requested resources are included in the provided VCs. Related to that, by adopting the JWT encoding for the VCs and by relying on existing standards, our solution can leverage a plethora of existing tools that perform most of the tasks required by the access control decision process.

By adopting the ZKP-based approach for implementing selective disclosure, SelectShare provides fine-grained access control, preserving at the same time the context of the output data. For example, in a solution

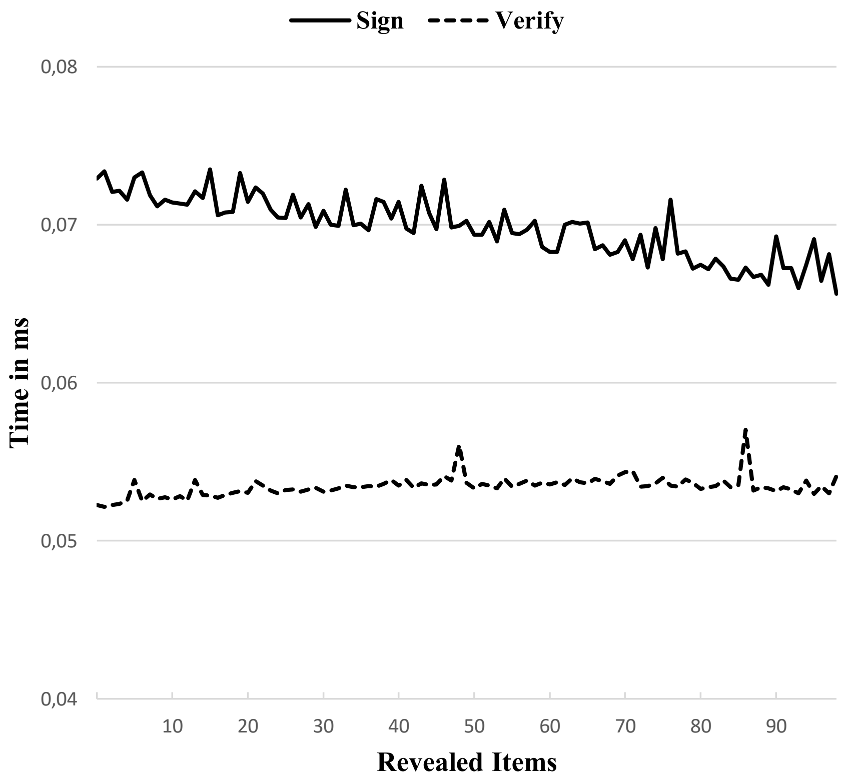

Fig. 2. Time to calculate a ZKP as a function of the revealed items

where each “field” in a JSON file is individually signed, additional measures must be considered in order to prevent a proxy from creating fake items by “combining” fields from different files.

5 Conclusions

In this paper we presented the design and implementation of SelectShare: an access control solution that allows fine-grained access control for IoT data sharing. SelectShare’s core components are built by leveraging already standardized solutions, which facilitates its integration with existing systems. Additionally, many security mechanisms of SelectShare are implemented in a HTTP proxy, hence, existing HTTP-based resources can be transparently protected. SelectShare facilitates interoperability and improves security management.
Ongoing and future work in this area includes tighter integration of VC with the WoT gateway, e.g., the Thing Description generated by the WoT gateway can include “specifications” of the expected VCs. Additionally, our system can be extended to support other means of client authentication (most notably *Decentralized Identifiers*), selective disclosure of VCs (achieving this way the principle of least privilege), as well as support for advanced trust relationships (e.g., delegation of access rights).

Acknowledgments

The work reported in this paper has been funded in part by European Union’s Horizon 2020 research and innovation programme through subgrant *Selective IoT data sharing (SelectShare)* of project *NGI DAPSI*.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-taa. In: International Conference on Security and Cryptography for Networks, pp. 111–125. Springer, Heidelberg, DE (2006)
2. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Annual International Cryptology Conference, pp. 41–55. Springer, Heidelberg, DE (2004)
3. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong diffie hellman assumption revisited. In: International Conference on Trust and Trustworthy Computing, pp. 1–20. Springer, Heidelberg, DE (2016)
4. D. Fett et al.: OAuth 2.0 Demonstrating of Proof-of-Possession at the Application Layer (DPoP). Rfc draft (2020). URL https://datatracker.ietf.org/doc/draft-ietf-oauth-dpop/
5. Dimitrakos, T., Dilshener, T., Kravtsov, A., La Marra, A., Martinelli, F., Rizos, A., Rosetti, A., Saracino, A.: Trust aware continuous authorization for zero trust in consumer internet of things. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 1801–1812 (2020)

6. Fotiou, N., Siris, V.A., Polyzos, G.C., Kortesniemi, Y., Lagutin, D.: Capabilities-based access control for iot devices using verifiable credentials. In: SafeThings 2022. IEEE (2022)
7. Ghosh, N., Chandra, S., Sachidananda, V., Elovici, Y.: Softauthz: A context-aware, behavior-based authorization framework for home iot. IEEE Internet of Things Journal **6**(6), 10,773–10,785 (2019)
8. Goyal, G., Liu, P., Sural, S.: Securing smart home iot systems with attribute-based access control. In: Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, Sat-CPS '22, pp. 37–46. Association for Computing Machinery, New York, NY, USA (2022)
9. Hardt (ed.), D.: The OAuth 2.0 authorization framework. RFC 6749, IETF (2012)
10. Jones, M.: JSON Web Key (JWK). RFC 7517, IETF (2015). URL https://tools.ietf.org/html/rfc7517
11. Kaastra, M., Tinkler, S., Tuzlic, S., Abts, M., Griggiths, F., Allen, J.: New Energy Consumer. Report, Accenture (2022). https://www.accenture.com/sk-en/insights/utilities/new-energy-transition-demand
12. Kalos, V., Polyzos, G.: Requirements and secure serialization for selective disclosure verifiable credentials. In: International Conference on ICT Systems Security and Privacy Protection (IFIP SEC), vol. 648. Springer, Heidelberg, DE (2022)
13. Kovatsch, M., Matsukura, R., Lagally, M., Kawaguchi, T., Toumura, K., Kajimoto, K.: Web of Things Architecture. W3C Recommendation, W3C (2020). https://www.w3.org/TR/wot-architecture/
14. Longley, D., Sporny, M.: Revocation list 2020. Draft Community Group Report, W3C (2021). https://w3c-ccg.github.io/vc-status-rl-2020/
15. M. Sporny et al.: Verifiable credentials data model 1.1. W3C Recommendation, W3C (2022). https://www.w3.org/TR/verifiable-claims-data-model/
16. Reijsbergen, D., Maw, A., Dinh, T.T.A., Li, W.T., Yuen, C.: Securing smart grids through an incentive mechanism for blockchain-based data sharing. In: Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy, CODASPY '22, pp. 191–202. Association for Computing Machinery, New York, NY, USA (2022)
17. S. Kaebish and T. kamiya and M. McCool and V. Charpenay and M. Kovatsch: Web of Things Thing Description. W3C Recommendation, W3C (2020). https://www.w3.org/TR/wot-thing-description/
18. Schuster, R., Shmatikov, V., Tromer, E.: Situational access control in the Internet of Things. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, pp. 1056–1073. ACM, New York, NY, USA (2018)
19. Shafagh, H., Burkhalter, L., Hithnawi, A., Duquennoy, S.: Towards blockchain-based auditable storage and sharing of iot data. In: Proceedings of the 2017 on Cloud Computing Security Workshop, CCSW '17, pp. 45–50. Association for Computing Machinery, New York, NY, USA

20. Shakarami, M., Benson, J., Sandhu, R.: Blockchain-based administration of access in smart home iot. In: Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, Sat-CPS '22, pp. 57–66. Association for Computing Machinery, New York, NY, USA (2022)
21. Sikder, A.K., Babun, L., Celik, Z.B., Acar, A., Aksu, H., McDaniel, P., Kirda, E., Uluagac, A.S.: Kratos: Multi-user multi-device-aware access control system for the smart home. In: Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '20, pp. 1–12. Association for Computing Machinery, New York, NY, USA (2020)
22. Sikder, A.K., Babun, L., Celik, Z.B., Aksu, H., McDaniel, P., Kirda, E., Uluagac, A.S.: Who's controlling my device? multi-user multi-device-aware access control system for shared smart home environment. ACM Trans. Internet Things (2022)
23. Sun, Y., Yin, L., Sun, Z., Tian, Z., Du, X.: An iot data sharing privacy preserving scheme. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 984–990. IEEE (2020)
24. T. Lodderstedt et al.: OAuth 2.0 Rich Authorization Requests. Rfc draft (2022). URL https://datatracker.ietf.org/doc/html/draft-ietf-oauth-rar
25. Whitehead, A., Lodder, M., Looker, T., Kalos, V.: The bbs signature scheme (2022). Https://identity.foundation/bbs-signature/draft-bbs-signatures.html