SelectShare- Selective IoT data sharing

SelectShare Architecture

Authors	George C. Polyzos (polyzos@aueb.gr), George Xylomenos (xgeorge@aueb.gr), Iordanis Koutsopoulos (jordan@aueb.gr), Vasilios A. Siris (vsiris@aueb.gr), Nikos Fotiou (fotiou@aueb.gr), Anna Kefala (a.kefala@aueb.gr), Evgenia Faltaka (eugeniafaltaka@gmail.com), Iakovos Pittaras (pittaras@aueb.gr), Athina Katsari (ak@pleg.ma), Nikos Ipiotis (ni@pleg.ma), Spiros Chadoulos (sc@pleg.ma), Stratos Keranidis (stratos@mydomx.eu), Polychronis Symeonidis (pol@mydomx.eu)
Version	1.0
Project website	https://mm.aueb.gr/projects/selectshare

1 INTRODUCTION

Although the IoT is expected to generate vast amount of data, the potentials of these data are 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 that improve comfort, surveillance for security and safety, etc. These applications are in most cases siloed and the generated data is only used for the specific purposes of each application. Nevertheless, these data can be valuable for other stakeholders that can collect and analyse them to provide "over the top" services such as smart energy profiling and optimizations, consumption forecasting based on machine learning, recommendation services, visualization services, and many others.

In all these cases, 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".

SelectShare aims to deliver a platform for controlled sharing of IoT data. The SelectShare architecture enables collection of 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 three components. First, an IoT gateway that collects data from IoT devices and makes them available by following W3C's Web of Things specifications facilitating data discovery and data interoperability. Second, an OAuth 2.0-based Verifiable Credential (VC) issuing mechanism for generating self-contained, fine-grained access tokens, as well as a corresponding HTTP-proxy that acts as a Policy Enforcement Point (PEP), for controlling access to the IoT gateway. Third, a data transcoder that transforms IoT data using a common JSON schema, supporting digital signatures based on Zero-Knowledge Proofs (ZKPs). These signatures are used to prove the integrity, authenticity, and provenance of the desired subsets of the original IoT data, while keeping sensitive data attributes private. Hence, the system provides to data consumers guarantees about data provenance and integrity while also preserving anonymity and revealing no sensitive information.

2 PROVIDED FUNCTIONALITY

The SelectShare architecture is an access control solution that relies on widely used standards. It provides **Security and Privacy** by implementing VC-based fine-grained access control and ZKP-based selective data disclosure, and **IoT Data Interoperability and Compatibility**, by supporting the WoT TD specifications. In the rest of this section, we detail the technology that enables these functionalities.

2.1 VC-based Access Control

SelectShare implements fine-grained access control, as well as the *principle of least privilege* by leveraging Verifiable Credentials (VCs).

A VC [1] allows an *issuer* to assert some *attributes* of a *subject*. A VC includes information about the issuer, the subject, the asserted attributes, as well as possible constrains (e.g., expiration date). Then, *holders* of a VC (which in most cases are the same entity as the VC subject) can prove to a *verifier* that they possess a VC with certain characteristics. To facilitate interoperability, the VC data model allows different VC *types* that define the attributes a VC should include.

SelectShare uses a VC type named *CapabilitiesCredential* defined by MMlab/AUEB. A VC of this type includes a property named *capabilities* that expresses the resources that a VC subject can access. This property includes pairs of resource names and allowed operations. This credential type is defined in the *context* https://mm.aueb.gr/contexts/capabilities/v1.

VCs in SelectShare are encoded as JSON Web Tokens (JWT) which include the following claims

- **iss**: The URL of the issuer.
- **cnf**: The public key the VC subject encoded as a JSON Web Key (JWK)
- **aud**: The URL of the target IoT gateway
- **nbf**: A timestamp before which the VC is not valid.
- **exp**: A timestamp indicating VC's expiration time.
- **vc**: A composite claim that describes the actual capabilities granted, which includes the following properties:
    - **type**: The type of the credential (i.e., CapabilitiesCredential).
    - **capabilities**: The capabilities property.

Finally, each VC is embedded in a JSON Web Signature (JWS) that can be verified using the public key of the issuer.

2.2 Data Interoperability

The SelectShare architecture provides IoT data interoperability by implementing an IoT gateway that abides by Web of Things (WoT) Things Description (TD) [2] 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 are made available through REST-based APIs, which can be used to access the device's *properties*, to trigger device *actions*, as well as to receive device-generated *events*.

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. All these “interaction affordances”' (a term coined by the WoT Architecture specification) are stored in tuples that map an affordance *identifier* to the corresponding access information.

2.3 Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs) are a fundamental notion in cryptography, in which an entity (the prover) proves knowledge of a piece of information that satisfies a certain relationship (for example, knowledge of a discrete logarithm), to another entity (the verifier), without revealing any information about the piece of information itself. This functionality, combined with the inherent composability of ZKPs has led to the creation and implementation of many cryptographic protocols using ZKPs.

One of those protocols is BBS+. BBS+ is a multi-message digital signature protocol, that also encapsulates ZKPs for a critical part of its functionality. It was first envisioned in [3] (from where it takes its name), touched again in [4], re-visited in [5] and is currently under standardization [6] . BBS+ can be thought as a composition of two (interdependent) components; the digital signature and the ZKP protocol. As a digital signature, BBS+ provides the ability to sign an array of individual messages (each message consisting of a string of octets), with only a single constant size signature. The signature can be validated given the signer's Public Key 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.

As a ZKP protocol, BBS+ enables any entity that knows the signature and the original signed array of messages, to 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-disclosed messages. The proof can be validated with only the signer's PK and the array of revealed messages. The whole protocol is zero-knowledge in the sense that, from this interaction, no information can be derived about the signature or the un-disclosed messages.

3 SPECIFICATIONS

In this section we detail the SelectSchare architecture (also illustrated in Figure 1) based on the provided functionalities.

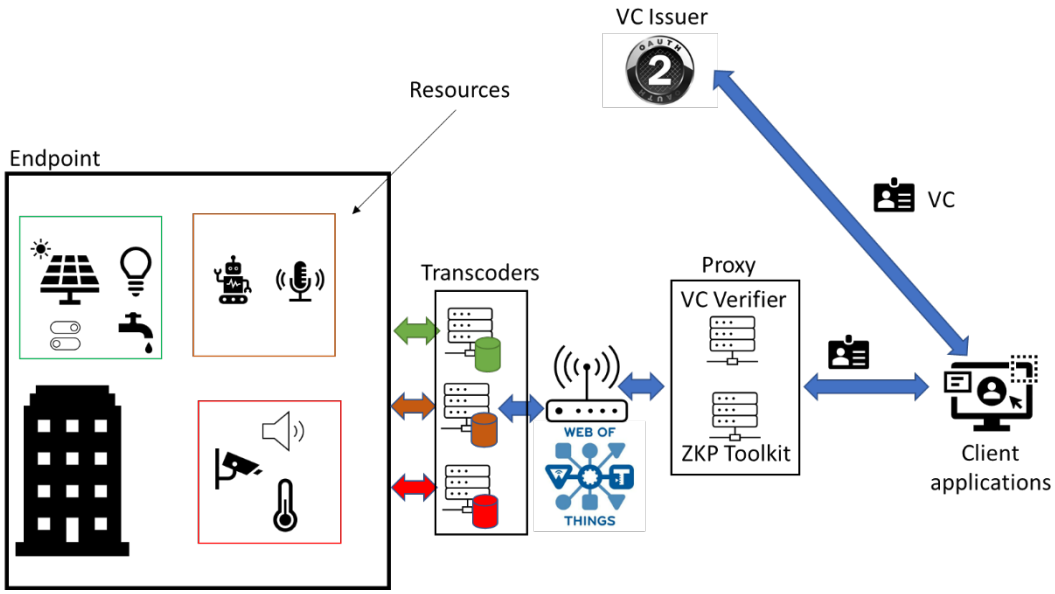

Figure 1 Overview of the SelectShare architecture

3.1 VC-based Access Control

SelectShare's VC-based access control functionality is implemented by the following modules:

- VC issuer
- VC verifier

The **VC issuer** is an OAuth 2.0 authorization server extended with VC issuing capabilities. Issued VCs are encoded as JWTs and signed using JWS, improving compatibility and integration with existing tools. SelectShare considers VCs that describe the **capabilities** of a client over a protected resource. Additionally, the SelectShare VC issuer maintains a VC revocation list.

The VC verifier is an HTTPS proxy that intercepts the communication between a client and an HTTP(S)-based **protected resource**. The VC verifier is able to verify the validity, the status, and the ownership of a VC. Additionally, the VC verifier acts as a **policy enforcement point** by validating whether or not a VC can be used for executing a particular request over a protected resource.

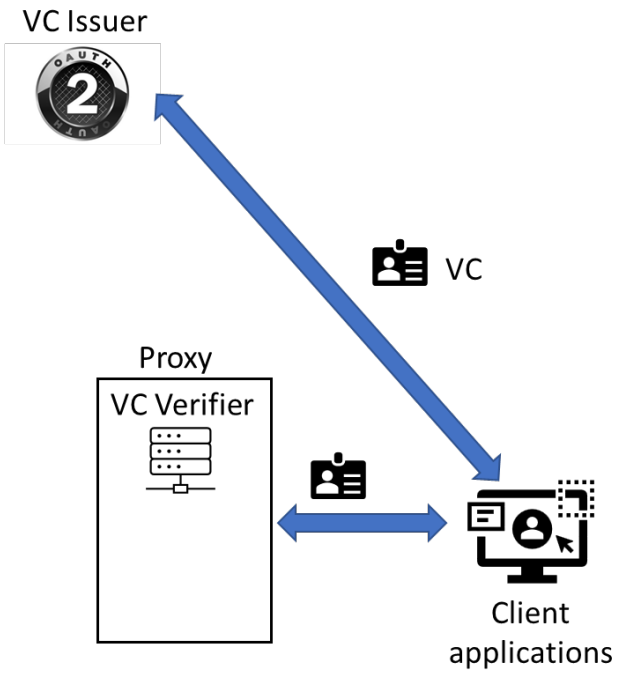

Figure 2 Modules implementing VC-based access control

A typical flow in SelectShare includes the following steps:

3.1.1 VC Issuer configuration

This is a step usually executed during a set up phase. With this step an issuer is configured with policies that specify the **capabilities** that correspond to a client. Clients are identified using a username and a password. Hence, the VC issuer maintains a data structure that maps usernames to lists of capabilities.

3.1.2 VC request and issuance

With this step, a client application requests from the issuer a VC. A VC request is in essence an OAuth 2.0 access token request using the client 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. The client proves possession of the corresponding public key using OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP).

The following request in example of a VC issuance request:

```
POST /issue HTTP/1.1
Host: <Issuer URL>
Content-Type: application/x-www-form-urlencoded
DPoP: <DPoP>
grant_type=client_credentials
```

A response to a successful request has the following form:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

```
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"<VC>",
  "token_type":"VC+JWT",
  "expires_in":<Seconds>,
}
```

A **VC** is the base64url encoding of a JWT singed by the Issuer, as specified by the VC data model. The generated JWT includes a **cnf** field, as specified by RFC 7800 that contains the public key of the client.

Τηε VCs used in SelectShare are of type "CapabilitiesCredential" defined in the context https://mm.aueb.gr/contexts/capabilities/v1. This type includes an array, called "capabilities", and each element of this array is a map that maps a "Resource" to a list of "capabilities". An example of a VC before encoding follows (the signature part is omitted).

```
{
  "typ": "jwt",
  "alg": "EdDSA"
}.
{
  "jti": "https://zero.corp/credentials/1",
  "iss": "https://zero.corp",
  "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": {
        "Device1": [
          "Read_Temperature",
          "Read_Humidity"
        ]
      }
    }
  }
}
```

3.1.3 Resource request

A client application requests an HTTP resource by including in its request the received JWT-encoded VC and 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). The request is received by the verifier that acts as an HTTP proxy. An HTTP request must include the **Authorization** header set to **DPoP** followed by the base64url encoded VC, and the **DPoP** header.

```
GET /<resource> HTTP/1.1
Host: <resource URL>
Authorization: DPoP <VC>
DPoP: <DPoP proof>
```

A DPoP proof used for requesting a resource in SelectShare includes the following fields (the signature is omitted):

```
{
  "typ": "dpop+jwt",
  "alg": "EdDSA",
  "jwk": {
      "kty": "OKP",
      "crv": "Ed25519",
      "x": "<Ed25519 public key>"
  }
}.
{
  "jti": <96bits pseudorandom>,
  "htm": "<HTTP method>"
  "htu": "<Issuer URL>",
  "iat": <Creation time>,
  "ath":"<base64url encoded SHA-256 hash of the VC>"
}
```

The verifier initially validates the included VC and DPoP proof. Then it examines the status of the VC by communicating with the VC issuer, using Revocation List 2020 [7]. In particular, the issuer maintains a revocation list that concerns all VCs it has issued. This list is a simple bitstring and each credential is associated with a position in the list. Revoking a VC means setting the bit corresponding to the VC to 1. Furthermore, each generated VC includes a field named "revocationListIndex" that specifies the position of the credential in the revocation list. The revocation list can be downloaded by performing an HTTP GET to this interface.

```
GET /status HTTP/1.1
Host: <Issuer URL>
```

The revocation list is also a JWT-encoded credential. E.g.,

```
{
  "typ": "jwt",
```

```
  "alg": "EdDSA"
}.
{
  "iss": "https://mm.aueb.gr/as",
  "iat": 1617559370,
  "exp": 1618423370,
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://w3id.org/vc-revocation-list-2020/v1"
    ],
    "type": ["VerifiableCredential",
"RevocationList2020Credential"],
    "credentialSubject": {
      "type": "RevocationList2020",
      "encodedList": "H4sIAAAAAAAAA-3BMQEAAADCo…wM92tQwAAA"
    }
  }
}
```


Finally, the verifier, verifies the included VC based on a set of **verification rules** defined by the resource owner during the set-up phase. If all checks succeed, the verifier forwards the request to the protected resource. In particular, the VC verifier examines if the user is authorized to access the “field” of the “deviceID” includes in the request URL: this is simply implemented by examining if these two parameters are included in the provided VC. In order to achieve this functionality, the corresponding filtering rule of the VC Verifier configuration file looks like this:

```
"filters" :[
     ["$.vc.credentialSubject.capabilities.#deviceID[*]", "#field"]
 ]
```

3.2 Data Interoperability

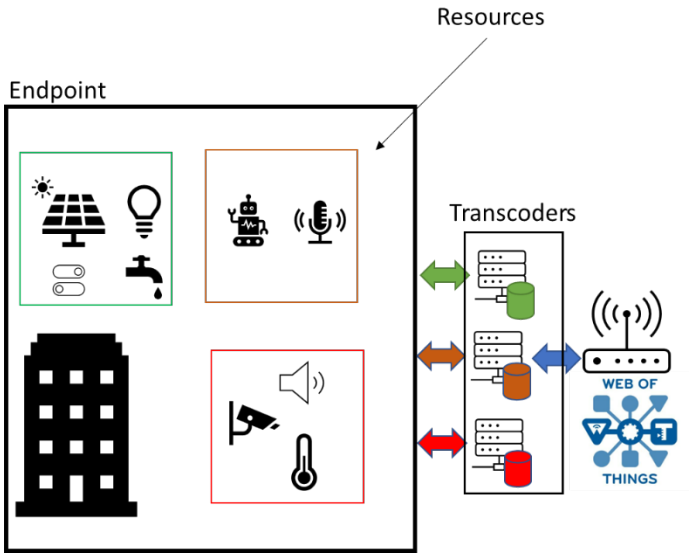

Figure 3 Modules implementing Data Interoperability

SelectShare provides a universal interface for accessing IoT resources that may use their own data model. This is achieved by using data Transcoders and a WoT Gateway (Figure 3). The role of a transcoder is to transform data received from an IoT device into JSON objects that follow a schema defined by the SelectShare architecture. The following listing includes the used schema:

```
{
  "$id": "https://mm.aueb.gr/contexts/capabilities/v1/schema.json",
  "$schema": "https://mm.aueb.gr/contexts/capabilities/v1",
  "type": "object",
  "properties": {
    "deviceID": {
        "type": "string"
      },
    "measurements": {
      "type": "array",
      "items": {"$ref": "#/$defs/measurement"}
    }
  },
  "$defs": {
    "measurement": {
      "type": "object",
      "properties": {
        "field": {
          "type": "string"
        },
        "values": {
          "type": "array",
          "items": {"$ref": "#/$defs/value"}
```

```
        }
      }
    },
    "value":{
        "type": "object",
        "properties": {
          "time": {
            "type": "string"
          },
          "value": {
            "type": "string"
          }
        }
     }
  }
}
```

SelectShare's WoT gateway provides a universal API for accessing IoT data measurements. This API is specified using WoT TD specifications. The current TD of the SelectShare WoT is included in the following listing:

```
{
  "@context": "https://www.w3.org/2022/wot/td/v1.1",
  "@type": "Thing",
  "security": [
    "nosec_sc"
  ],
  "properties": {
    "device": {
      "type": "object",
      "readOnly": true,
      "uriVariables": {
        "deviceID": {
          "type": "integer"
        },
        "field": {
          "type": "string"
        },
        "startTime": {
          "type": "string"
        },
        "endTime": {
          "type": "string"
        }
      },
      "forms": [
        {
          "href":
"http://:8080/building01/properties/device{?deviceID,field,,startT
ime,endTime}",
          "contentType": "application/json",
```

```
          "op": [
            "readproperty"
          ],
          "htv:methodName": "GET"
        }
      ],
      "writeOnly": false,
      "observable": false
    }
  },
  "title": "Building01",
  "description": "SelectShare sample WoT",
  "id": "urn:uuid:727dddd4-a207-401d-9e3e-93e60d0a6bd3",
  "forms": [
    {
      "href": "http://10.0.2.15:8080/building01/properties",
      "contentType": "application/json",
      "op": [
        "readallproperties",
        "readmultipleproperties"
      ]
    }
  ],
  "securityDefinitions": {
    "nosec_sc": {
      "scheme": "nosec"
    }
  }
}
```

3.3 Selective Disclosure using ZKPs

SelectShare uses BBS+ signatures to enable selective disclosure of IoT data. Recall that the IoT Data have been encoded as a JSON object and signed by transcoders. Selective disclosure involves three algorithms: data framing, canonicalization, and ZKP generation.

3.3.1 Data 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 (not the values) that lead to the values that the prover will want to reveal. For example, using as input the following JSON item:

```
{
    "measurements": {
        "temperature":"30oC",
        "humidity":"60%"
```

```
    }
}
```

And the following frame:

```
{
    "measurements": {
        "temperature":"",
    }
}
```

The framing algorithm will output the following sub-item

```
{
    "measurements": {
        "temperature":"30oC",
    }
}
```

The framing algorithm used in SelectShare is inspired by the framing technique introduced and used for JSON Linked Data (JSON-LD), but simplified and adapted to work on JSON-encoded items.

3.3.2 Canonicalization

As discussed previously, BBS+ signatures act on arrays of messages and not on structured data formats like JSON. In order for an owner to be able to sign a data item, as well as in order for a storage node 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 [8] which is a lightweight, provably secure, JSON canonicalization proposal, designed to work with any data model.

3.3.3 Proof generation

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+.

3.4 Selective disclosure in SelectShare

The function of selective disclosure is implemented in a distributed manner by the transcoder and the proxy module (see also Figure 4). 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 HTTP proxy is responsible for framing the signed object and for generating the corresponding ZKP. The framing operation is implemented by taking into consideration the "field" option included in the request URL. 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 by the VC verifier.

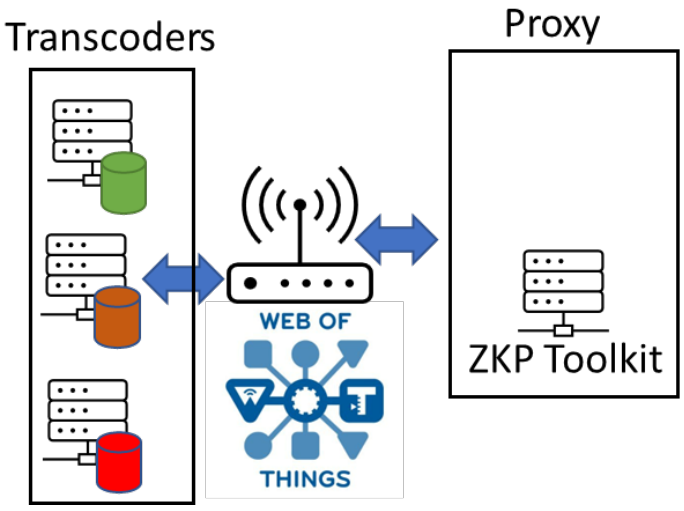

Figure 4 Modules implementing selective disclosure using ZKPs

4 REFERENCES

[1] Verifiable Credentials Data model v1.1, available at https://www.w3.org/TR/vc-data-model/

[2] Web of Things (WoT) Thing Description, available at https://www.w3.org/TR/wot-thing-description/

[3] Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short Group Signatures. In Annual International Cryptology Conference. Springer, Heidelberg, DE, 41–55

[4] Man Ho Au, Willy Susilo, and Yi Mu. 2006. Constant-Size Dynamic k-TAA. In International Conference on Security and Cryptography for Networks. Springer, Heidelberg, DE, 111–125.

[5] Jan Camenisch, Manu Drijvers, and Anja Lehmann. 2016. Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited. In International Conference on Trust and Trustworthy Computing. Springer, Heidelberg, DE, 1–20

[6] Andrew Whitehead, Mike Lodder, Tobias Looker, and Vasilis Kalos. 2022. The BBS Signature Scheme. https://identity.foundation/bbssignature/draft-bbs-signatures.html

[7] WCC Group. (2020) Revocation list 2020. [Online]. Available: https://w3c-ccg.github.io/vc-status-rl-2020/

[8] V. Kalos, G.C. Polyzos, "Requirements and Secure Serialization for Selective Disclosure Verifiable Credentials", in IFIP Sec 2022