decode

IoT privacy-enhancing data sharing: integration with Pilot Infrastructures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732546

Project no. 732546

DECODE

DEcentralised Citizens Owned Data Ecosystem

[D3.9] IoT privacy-enhancing data sharing: integration with pilot infrastructures

Version Number: V1.0

Lead beneficiary: Thingful

Due Date: 31st May 2019

Author(s): Sam Mulube (THINGFUL)

Editors and reviewers: Javier Rodriguez (IMI BCN), Oleguer Sagarra Pascual (DRIBIA), Andrea D'Intino (DYNE.ORG), Rohit Kumar (EURECAT)

Dissemination level:		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Approved by: Francesca Bria (Chief Technology and Digital Innovation Officer, Barcelona City Hall)

Date: 31/05/2019

This report is currently awaiting approval from the EC and is not considered to be a final version.

1 Contents

1.1 Figures

1.2 Tables

2 Introduction

Earlier work within DECODE performed by Thingful [1] in association with the other consortium partners involved the creation of a suite of cooperating microservices that attempted to provide a robust architecture for IoT devices but one in which the normal structures of control were inverted. Rather than the provider of the device or the IoT infrastructure being in control of the storage and transmission of data, we attempted to give that control to the users generating the data.

The mechanism used for this inversion was based on applying strong encryption techniques to the data generated by devices and giving control of that encryption to the user rather than the relying on the infrastructure. This "cryptographic control" allows DECODE to sidestep the philosophical question of whether data can ever be owned, by instead framing the problem as being "who controls the keys used to encrypt the data?". With this framing it doesn't matter who gains access to the raw data being emitted from the device as only the specific entities for whom the data is encrypted will be able to decode and make sense of it.

More detail on this scheme can be found in deliverable D3.7 [2] which attempts to document the previous work that Thingful did in this project, however the purpose of this document is to detail the additional work performed by Thingful to develop an additional software component which aims to become the primary user facing interface that citizens will use to participate in the IoT pilot.

A mobile app (called hereafter the DECODE app) was already developed for the related DDDC [3] pilot by Dribia [4]. The consortium decided to improve this mobile app to incorporate a range of extended functionalities and make it fully modular, extensible and ready for deployment by third parties, however it was realised that this work would take too long in relation to the schedule already planned for the IoT pilot workshops.

In order to demonstrate and make real the IoT pilot by integrating the already developed components into a fully functioning system, Thingful agreed to develop an additional software component (hereafter referred to as the DECODE webapp) that would take the place of the DECODE app for the entirety of the IoT pilot and it is this component that we describe in this document.

2.1 Partners

2.1.1 Dribia

Dribia [4] were responsible for building the current DECODE app that served as the inspiration for the component described in this document, but beyond that they also played a crucial role in helping design the IoT pilot functionalities and experience, as well as supporting IMI in the coordination and management of the project and working with IMI to plan and organise the community engagement sessions.

2.1.2 Dyne.org

Dyne.org [5] are the technical lead partners for the IoT pilot, and in addition to this role they also have provided a tool known as Zenroom that provides the core cryptographic engine that is used for both securely encrypting the IoT data that flows through the pilot, but also providing an implementation of the Coconut [6] protocol by which participants can share attributes with others in a cryptographically secure way. The credential issuing service described later in this document also uses Zenroom to generate credentials that enable this interaction.

2.1.3 Eurecat

Eurecat [7] are the partner that have been developing the BCNNow dashboard where all data from the Barcelona pilots is displayed. They have also been a key partner in helping shape the experience, and have built tools to integrate the dashboard application with the components previously developed by Thingful.

2.1.4 IMI

L'Institut Municipal d'Informàtica (IMI) [8] are the partner who have played the role of the primary project management and coordination point to ensure the various partners developing the components described were able to work together efficiently to deliver a functional system for the pilot. IMI also worked with Dribia and SmartCitizen to organise the community engagement sessions. Without IMI's work in this area the project would not have been possible in the way that it has been realised.

2.1.5 SmartCitizen

SmartCitizen [9] are not formal partners in the consortium however they have played a crucial part in its development. SmartCitizen have an existing infrastructure of IoT devices and data infrastructure that we were able to utilise in order to demonstrate the novel functionality described in D3.7 [2]. In addition to providing this technical underpinning, SmartCitizen also have expertise in community engagement projects and have been working with IMI and Dribia in order to plan and execute community engagement events where participants were found who wanted to take part in the pilot.

2.1.6 Thingful

Thingful [1] are the consortium partner responsible for developing the software component described in this deliverable with the help and support of the other consortium partners. Prior to this work Thingful also developed the core infrastructure that was designed and developed for the IoT pilot. These components are briefly described in the later sections of this report.

3 Requirements and Analysis

3.1 Document Scope

The purpose of this document is to document the additional work performed by Thingful to develop a user facing component that is the primary interface citizens taking part in the IoT pilot will interact with. All the previous software components developed by Thingful for the pilot are explicitly back-end server components whose only interface is the command line (when running the applications), or an HTTP based API that interacts only using Protocol Buffers or JSON.

Within that context we aim to cover the following areas:

- The basic rationale for the component to be developed
- Any requirements and constraints we identified for the application
- Description of the proposed user journeys when using the application from onboarding with SmartCitizen to accessing collected data via the Barcelona Now dashboard.
- Overall system conceptual and architectural design
- Description of the final component as delivered to partners

In addition, we will also include a brief section describing the deployment environment of this and the other three components previously developed for D3.7 [2] as these details were not fully available at the time of completing that deliverable.

Out of scope for this document are detailed descriptions of the components previously developed and delivered other than any detail needed to place the project described here in their proper context.

3.2 Constraints

We identified the following constraints which pertained to development of the application:

1. Development of the application was under severe time constraints – the original plan was to extend the existing DECODE app, but when the consortium decided to modify the roadmap, we only had a few months within which to develop a replacement for that app that offered equivalent functionality.

3.3 Requirements

We also identified the following high-level requirements:

1. The application must be able to interact with the software components previously developed by Thingful (Policystore and Encoder services), as well as the Credential issuing service developed by Dyne.org [5]. This simply means that it must be able to make HTTP requests and be able to process and work with JSON data.
2. The application must have programmatic access to a recent build of Zenroom [10] as we use Zenroom to provide the cryptographic operations by which users can create credentials. These credentials allow participants to securely and privately prove they have the right to access certain datasets.
3. The application should have similar UX to the DECODE app developed by Dribia for the DDDC [3] pilot and be primarily designed for use on mobile phones.

4. The application should not require any sort of backend data store, meaning all data must be stored locally within the application, and this data should be persistent, so users are able to configure devices and credentials and be able to return to the application later and be able to recreate the previous state of the application.

The previous application was developed as a native mobile app using a framework called React Native [11]. This was a technology that Thingful had no specific expertise in, however in looking at the high-level requirements described above we realised there was no specific requirement for developing a native mobile app, so we decided that it would be enough to develop a simple web application that is designed to replicate the mobile app experience. This decision allowed Thingful to quickly develop the application and so de-risk the development process.

3.4 Existing Software Components

The following components were previously created by Thingful, Eurecat, Dyne.org and SmartCitizen with the support, design participation and guidance of Dribia.

The purpose of the application described in this document is to provide the bridge between the components described below that orchestrates their functionality to create the experience designed by the consortium partners for the IoT pilot.

3.4.1 Encrypted Datastore

The Encrypted Datastore is a component developed by Thingful which saves the encrypted data generated by provisioned devices after they have been added to DECODE. This component exposes a simple read/write API where data can be read keyed by the community identifier with additional parameters that allow specific time windows to be requested. The software was developed using the Go [12] programming language and is currently being hosted by SmartCitizen on their system infrastructure.

The service has been deployed to https://datastore.decodeproject.eu/, and documentation of its API is available here: https://decodeproject.github.io/iot-datastore-docs/. The source code for this component is available here: https://github.com/decodeproject/iotstore.

3.4.2 Stream Encoder

The Stream Encoder is a component developed by Thingful which subscribes to SmartCitizen's MQTT broker. It exposes a read/write API that allows streams to be created or deleted. When a stream is created the encoder subscribes to the broker and starts collecting and encrypting and processing incoming data events. The software was developed using the Go [12] programming language and is currently being hosted by SmartCitizen on their system infrastructure.
The service has been deployed to https://encoder.decodeproject.eu/, and documentation of its API is available here: https://decodeproject.github.io/iot-encoder-docs/. The source code for this component is available here: https://github.com/decodeproject/iotencoder.

3.4.3 Policystore

The Policystore is a component developed by Thingful that exposes an API which allows community policies to be created or deleted. It also exposes a method by which clients can request a list of the currently active community policies which can then be applied to their devices. The software was developed using the Go [12] programming language and is currently being hosted by SmartCitizen on their system infrastructure.

The service has been deployed to https://policystore.decodeproject.eu/, and documentation of its API is available here: https://decodeproject.github.io/iot-policystore-docs/. The source code for this component is available here: https://github.com/decodeproject/iotpolicystore.

3.4.4 Credential Service

The Credential service is a component developed by Dyne.org which is a crucial part of the Coconut credential scheme. It exposes an API by which authorizable attributes can be created, and blind proof credentials can be created and verified. This service has been written as a Python [13] web application and is currently being hosted by Dyne.org on their system infrastructure.

The service has been deployed to https://credentials.decodeproject.eu/, and documentation of its API is available here: https://credentials.decodeproject.eu/docs . The source code for this component is available here: https://github.com/DECODEproject/dddc-credential-issuer.

3.4.5 Barcelona Now tool

The Barcelona Now (BCNNow) tool is the service that exposes a set of visualisation tools where the IoT data collected from participant devices will be visualised after being collected. It exposes an API for creating new private dashboards for the API pilot and plays a crucial part in the Coconut credential service. It has been developed as a Python [13] web application and is currently deployed on Eurecat's system infrastructure.

The application has been deployed to http://bcnnow.decodeproject.eu/ and documentation of its API is available here: https://decodeproject.github.io/bcnnow-dashboard-docs/. The source code for this component is available here: https://github.com/DECODEproject/bcnnow.

3.4.6 SmartCitizen Onboarding Application

The SmartCitizen Onboarding application is a component that has been developed by SmartCitizen as an extension of their existing onboarding flow. It provides a wizard that guides users through the complex process of configuring an IoT device to connect to their network and have it start collecting data.

The application has been deployed to https://start.decode.smartcitizen.me and is running on SmartCitizen's system infrastructure.

3.4.7 SmartCitizen MQTT Broker

Data published by devices connected to the SmartCitizen's data infrastructure publish data events via an MQTT broker from which the stream encoder can subscribe and start receiving events.

The broker has been deployed to mqtts://mqtt.smartcitizen.me and is running on SmartCitizen's system infrastructure.

4 System Design

4.1 User Journeys

The application is intended to be the primary interface by which pilot participants can take an active part in the pilot. To do this the application is required to be the bridge between an existing onboarding workflow developed by SmartCitizen and the BCNNow visualization tool developed by Eurecat.

To do this the application was required to provide functionality to implement the following two primary user journeys.

For more details on the terms used in the remainder of this document please see previous deliverables written by IMI and Thingful:

- D5.6 Deployment of Pilots in Barcelona [14]
- D3.7 Control and entitlement system for data owners [2]

4.1.1 Device Onboarding

The starting point in the process is the procedure called *Onboarding*. The term onboarding as described in this document is the procedure by which a new user can be given a physical device, and guided through a workflow which a) configures the device in order to allow it to connect to the user's home network, and b) captures some metadata about the device which places it in context (i.e. what is the geolocation of the device, is it situated indoors or outdoors, etc.)

To onboard a new IoT device into the DECODE pilot the user journey we designed collectively is as follows:

1. The user will first complete the existing onboarding flow already developed by SmartCitizen. This is a sophisticated application that guides users through the process of configuring the device to connect to the user's home WIFI and capturing some metadata about the physical location of the device (its geolocation, and the fact of whether the device is situated indoors or outdoors). The user will also be invited to choose a name for the device.
2. The onboarding application will create a QR code which encodes the above metadata as well as a unique code generated for the device which uniquely identifies it.
3. The user will then be prompted to scan this QR code using any third party QR code scanning app, and this will pass this information to the DECODE webapp.
4. If this is the first time the user has used the DECODE webapp, they will be prompted to enter a unique PIN which is the mechanism by which they can authenticate themselves to the DECODE webapp.
5. Once they have logged in with this PIN, the user will have the chance to review the device configuration information, and if it looks correct, they will be able to add this device configuration into the DECODE webapp.

A sequence diagram that attempts to capture the above workflow is shown in Figure 1 - Sequence diagram for user onboarding flow.

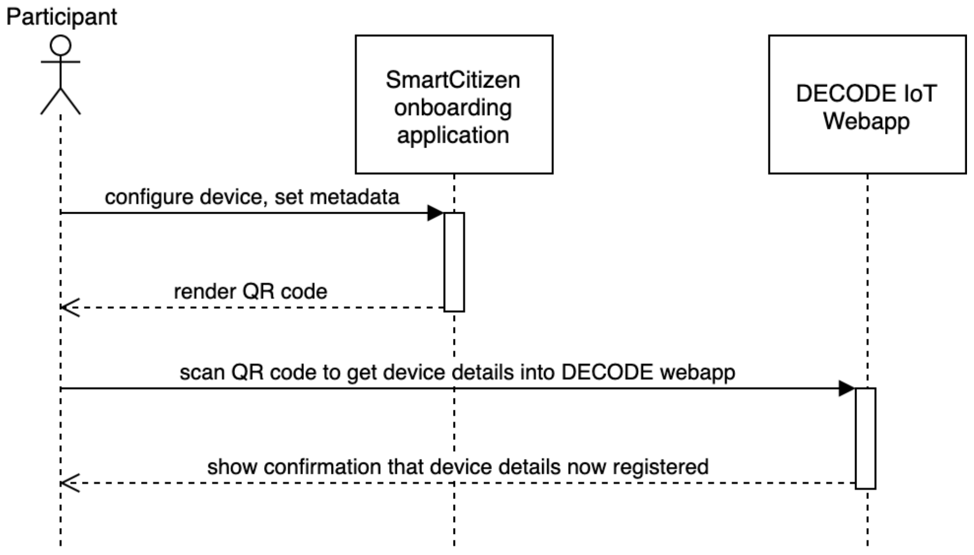

Participant
SmartCitizen onboarding application
DECODE IoT Webapp
configure device, set metadata
render QR code
scan QR code to get device details into DECODE webapp
show confirmation that device details now registered

Figure 1 - Sequence diagram for user onboarding flow

4.1.2 Join Community / Create Encrypted Stream

Once a device has been added the user will be guided to join a community which is the required next step for the user to collect data from their device.

1. The DECODE webapp should then show a page that prompts the user to select a community to join. These communities are defined by a policy definition which describes a set of operations on each of the data channels published by each device. These differential sets of operations are how we defined different groups with different privacy concerns.
2. Once the user selects a community, the DECODE webapp then needs to do two moderately complex operations:
    a. Create a cryptographically secure credential for the community which can be used later in order to allow the user to prove they have joined the community. This involves making a call to the credential service developed by Dyne.org and running Zenroom scripts in order to create a set of blind proof credentials.
    b. Create an encrypted stream which will start data being collected for the device, encrypted by the stream encoder and written to the encrypted datastore. This involves making a call to the stream encoder service.
3. The above complexity is hidden from the user however, so once they have joined the community the DECODE webapp should then show a confirmation message and save all the configuration into a persistent data store.

A sequence diagram that attempts to show the above workflow can be seen in Figure 2 - Sequence diagram for joining a community.

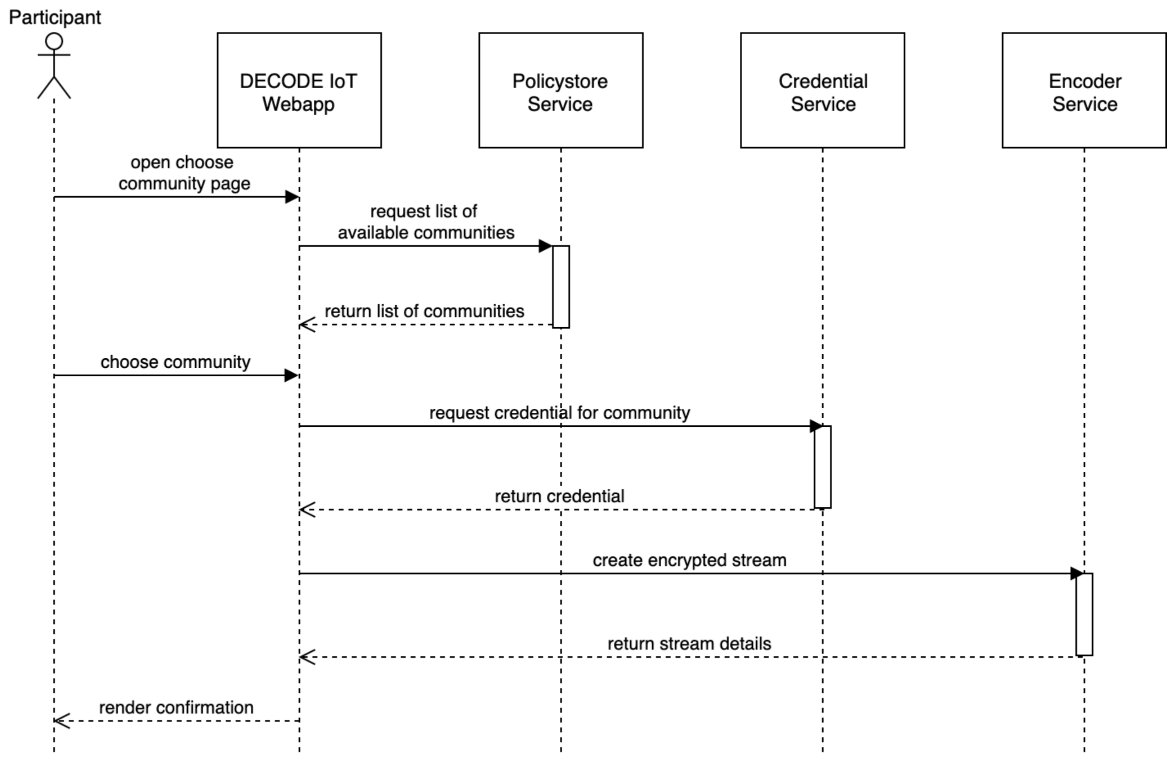


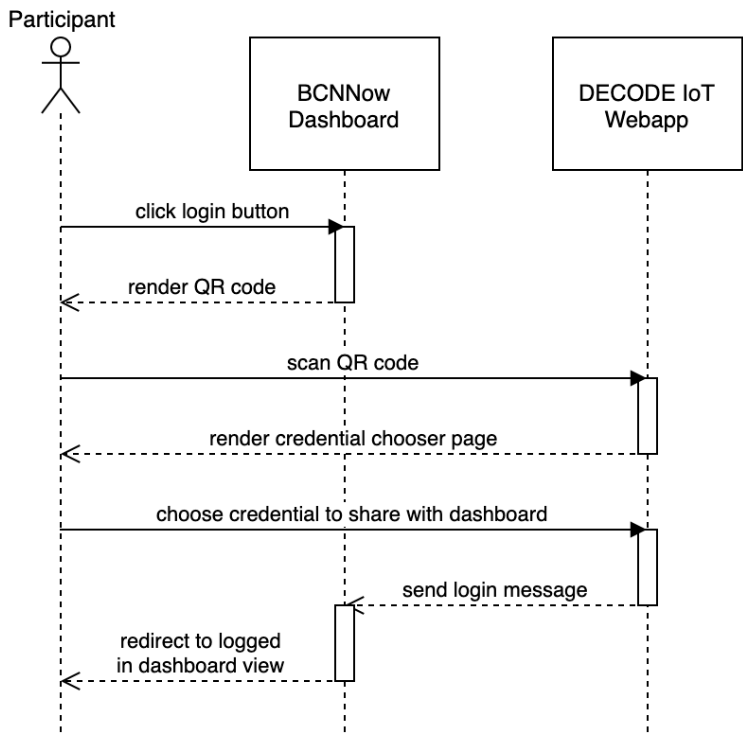
Figure 2 - Sequence diagram for joining a community

4.1.3 BCNNow Login

Once the user has created a stream and obtained a valid credential for the community, we designed the following workflow which is how the user is able to use that credential in order to prove to the dashboard that they should be granted access to a private dashboard.

1. The user will go to the Barcelona Now page and will click a button to login using the DECODE webapp.
2. The Barcelona Now tool should then render a QR code which encodes a URL that references the DECODE webapp, including a unique session identifier for the user.
3. The user will then be prompted to scan the above QR code with any third party QR code scanning app, which will open a page on the DECODE webapp which asks the user whether they would like to share a credential with the dashboard.
4. If the user agrees to share a credential, then the DECODE webapp will send back the chosen credential to the dashboard.
5. The dashboard application will then execute a Zenroom script to verify that the submitted credential is valid, and if this verification is successful the dashboard should then log the user in, redirecting the user's browser to a logged in view.
6. Finally, the DECODE webapp will show a confirmatory message confirming that the credential has been successfully shared.

A sequence diagram that attempts to show the above workflow can be seen in Figure 3 - Sequence diagram for logging into dashboard using webapp.



Participant
BCNNow Dashboard
DECODE IoT Webapp
click login button
render QR code
scan QR code
render credential chooser page
choose credential to share with dashboard
send login message
redirect to logged in dashboard view

Figure 3 - Sequence diagram for logging into dashboard using webapp

4.2 System Architecture

For a full explanation of the system architecture please refer to deliverable D3.7 [2] which aims to provide a comprehensive description of the overall system architecture; however, we include a copy of the system architecture here with the DECODE mobile app replaced with the DECODE webapp in Figure 4 - DECODE IoT System Architecture.

The only difference in the architecture from the schematic presented in D3.7 is the replacement of the DECODE app in the centre with the DECODE webapp described in this document.

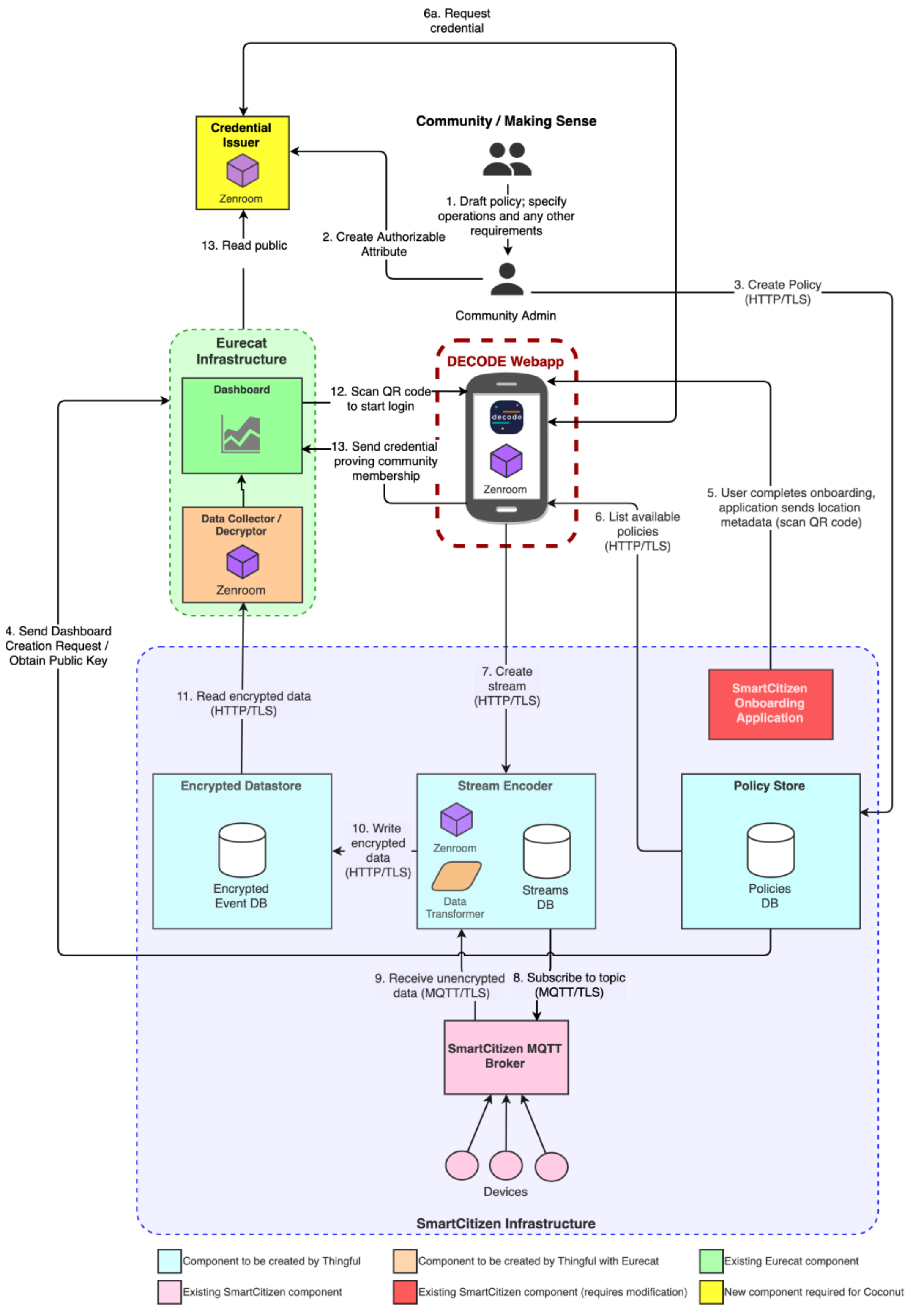

6a. Request credential
Credential Issuer
Zenroom
Community / Making Sense
1. Draft policy; specify operations and any other requirements
2. Create Authorizable Attribute
13. Read public
3. Create Policy (HTTP/TLS)
Community Admin
Eurecat Infrastructure
Dashboard
Data Collector / Decryptor
Zenroom
DECODE Webapp
Zenroom
12. Scan QR code to start login
13. Send credential proving community membership
6. List available policies (HTTP/TLS)
5. User completes onboarding, application sends location metadata (scan QR code)
4. Send Dashboard Creation Request / Obtain Public Key
11. Read encrypted data (HTTP/TLS)
7. Create stream (HTTP/TLS)
SmartCitizen Onboarding Application
Encrypted Datastore
Encrypted Event DB
10. Write encrypted data (HTTP/TLS)
Stream Encoder
Zenroom
Data Transformer
Streams DB
Policy Store
Policies DB
9. Receive unencrypted data (MQTT/TLS)
8. Subscribe to topic (MQTT/TLS)
SmartCitizen MQTT Broker
Devices
SmartCitizen Infrastructure
Component to be created by Thingful
Component to be created by Thingful with Eurecat
Existing Eurecat component
Existing SmartCitizen component
Existing SmartCitizen component (requires modification)
New component required for Coconut

Figure 4 - DECODE IoT System Architecture

5 Implementation

5.1 Technology Choices

It was clear from the start of this process that Thingful would be developing a web application rather than a mobile application, so given the limited time constraints available we decided to implement the application using the following technologies that Thingful has significant experience using.

5.1.1 Phoenix

Phoenix (https://phoenixframework.org) is a web application development framework created using a programming language called Elixir. Elixir is a functional language that is part of the wider Erlang ecosystem which takes advantage of the incredibly well tested and robust Erlang virtual machine (called BEAM) to deliver reliable and fault resistant development environment. Phoenix is a framework built on top of this foundation that provides a modern and fully featured web development environment.

5.1.2 Vue.js

Vue (https://vuejs.org) is a front-end JavaScript framework that allows for the easy building of complex single page applications with rich interactive behaviours. It provides tools for building frontends with multiple pages and handles all routing and parameter handling when navigating between pages. In addition, it provides a robust way of managing application state via a library called Vuex. By managing the entire front-end state via this library, we have a good way of handling the requirement of being able to save the application's state persistently allowing us to support this required functionality.

5.1.3 Browser Local Storage

We identified that for the web application to be a viable replacement for the mobile app we would need some mechanism for storing the application's state persistently. One option for this would have been to implement a persistence layer in the server backend; however, we really didn't want to do this as it would increase the complexity of the backend significantly.

The alternative we identified was to use something called browser `localStorage [15]`. This is an API offered by all modern web browsers which provides a domain scoped key value store, which persists beyond the scope of a user session. This storage mechanism is a simple key value store which allows string values to be saved. It does not permit the storing of arbitrary data types or objects; however, we can work around this by JSON encoding values before saving them to local storage, and then JSON decoding values when reading them back.

By using this storage area, we can save the user configuration state in such a way that after logging out and even closing the browser, the user will be able to restore that state after supplying their PIN.

5.1.4 Docker

As with the previous set of applications developed by Thingful we also decided to use Docker (https://www.docker.com/) as both a local development environment, and as the unit of delivery of the final application. This ability to ship our application to production using Docker is a great advantage as it allows the application to be easily deployed in multiple environments.

6 Results and Discussion

6.1 DECODE WebApp

6.1.1 Description

As mentioned in the previous chapter the DECODE webapp was developed as a Phoenix web application with a Vue.js front-end. The Phoenix backend serves the basic HTML framework within which the Vue.js application then runs, and in addition this backend exposes a websocket handler which accepts incoming messages from the front end and formats and sends these messages on to the correct external service. Responses from these external services are received by the Phoenix backend server, and then dispatched back to the front-end application over the websocket connection. An alternative approach to implementing this would have been to have the front-end JavaScript application directly calling external services however we chose this method as it simplifies the front-end to only send messages to one place (i.e. emitting messages over our websocket connection), and allows the back end to be the place that has knowledge of where those messages are ultimately supposed to be sent.

A high-level view of the application's internal system architecture is shown in Figure 5 - Internal architecture of the Phoenix webapp.

6.1.2 Deployment Location

The webapp is deployed and running at the following address: https://iot.decodeproject.eu

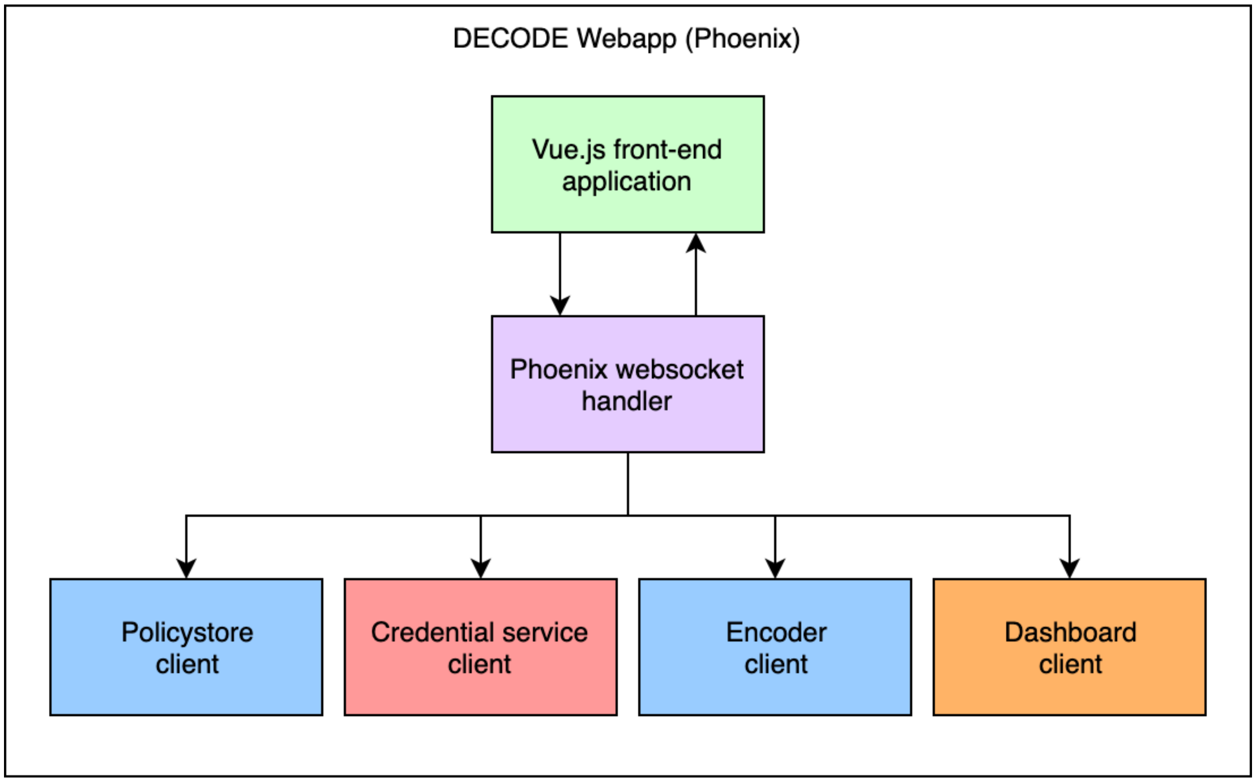

DECODE Webapp (Phoenix)
Vue.js front-end application
Phoenix websocket handler
Policystore client
Credential service client
Encoder client
Dashboard client

Figure 5 - Internal architecture of the Phoenix webapp

6.1.3 Source Repository

The source repository for the web application can be found at the following GitHub URL:

https://github.com/thingful/decodeweb

6.1.4 Docker Repository

The DECODE webapp developed by Thingful has been published as a container image and is available for download here: https://hub.docker.com/r/thingful/decodeweb.

6.1.5 Screenshots of the complete onboarding and access flows

For a reminder of the onboarding flow, please refer to the description and sequence diagram presented in section 4.1.1. To illustrate how this was implemented in practice we now include a series of screenshots which attempts to illustrate the application user interface seen by the user from first onboarding their SmartCitizen key using the SmartCitizen onboarding application, through choosing a community to add their device's data to, using the DECODE webapp to control this device, and then finally using the DECODE webapp to login to the BCNNow tool.

The process starts by the user being guided through a "wizard" style process on the SmartCitizen onboarding application that helps the user to configures the device's network connection and collects metadata from the user about where the device is located and whether the device has been placed indoors or outdoors. Once this data has been collected and the device has been properly configured, the SmartCitizen onboarding application constructs a URL containing the collected metadata as query parameter. This URL is encoded into a QR code and presented to the user.

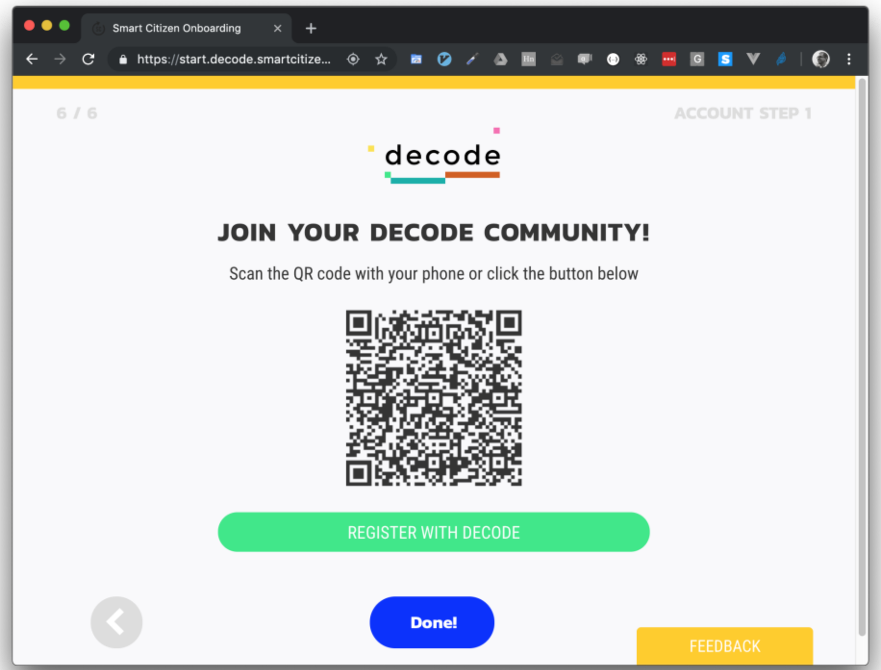
The parameters we agreed with SmartCitizen that should be encoded into the URL are shown in the table below:

Table 1 – SmartCitizen onboarding parameters

Parameter name	Description
`device_name`	A name given to the user during onboarding that allows them to identify their device when listed amongst others.
`device_token`	A unique code for the device generated during the onboarding process that identifies the specific device when installed in the user's home
`lng`	The longitude of the device expressed as a decimal value
`lat`	The latitude of the device expressed as a decimal value
`exposure`	Whether or not the device is situated indoors or outdoors

Every parameter shown is required, so must be submitted by the user. A screenshot of the final onboarding page showing the generated QR code is shown in Figure 6 - SmartCitizen onboarding application.

In this screenshot we can see the QR code along with a message encouraging the user to register with DECODE. If the user scans the QR code shown using any third-party QR code scanning app they are then taken over to the DECODE webapp to complete the onboarding workflow.

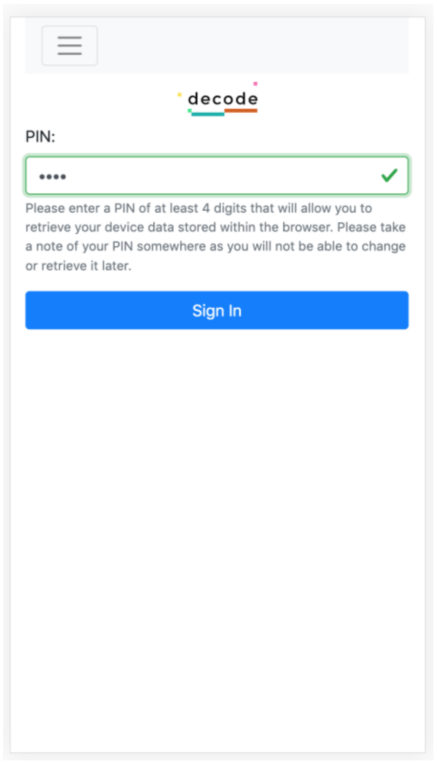


JOIN YOUR DECODE COMMUNITY!

Scan the QR code with your phone or click the button below

Figure 6 - SmartCitizen onboarding application

On scanning the QR code on their mobile device, the user will be taken to the following page on the DECODE webapp:

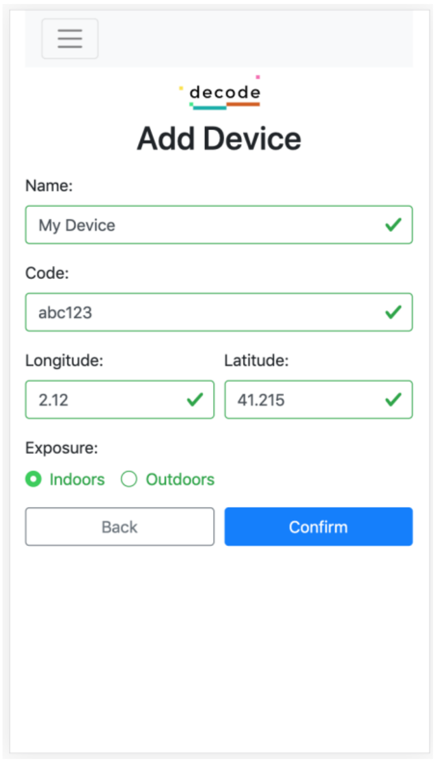

PIN:

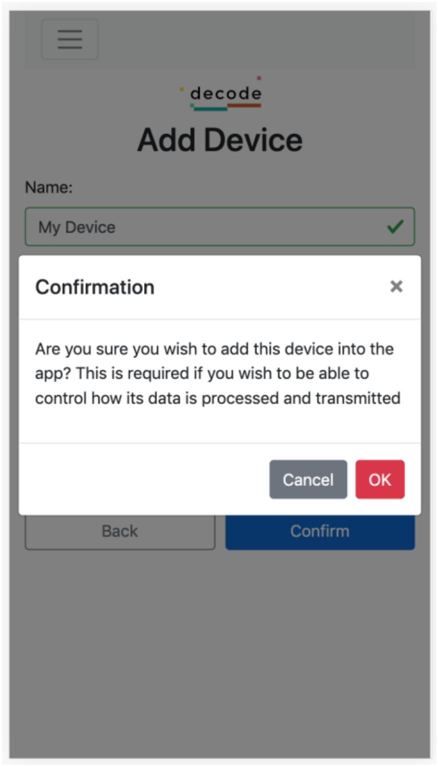
••••

Please enter a PIN of at least 4 digits that will allow you to retrieve your device data stored within the browser. Please take a note of your PIN somewhere as you will not be able to change or retrieve it later.

Figure 7 - DECODE webapp login page

The user is prompted to choose a PIN which is a simple mechanism by which we can protect the data stored within the browser on the participant's mobile device. After entering a PIN, the user will be immediately redirected to a page within which the application has extracted the parameters received from SmartCitizen and displays them to the user in order to allow them to confirm the details (shown in Figure 8 – DECODE webapp new device page).





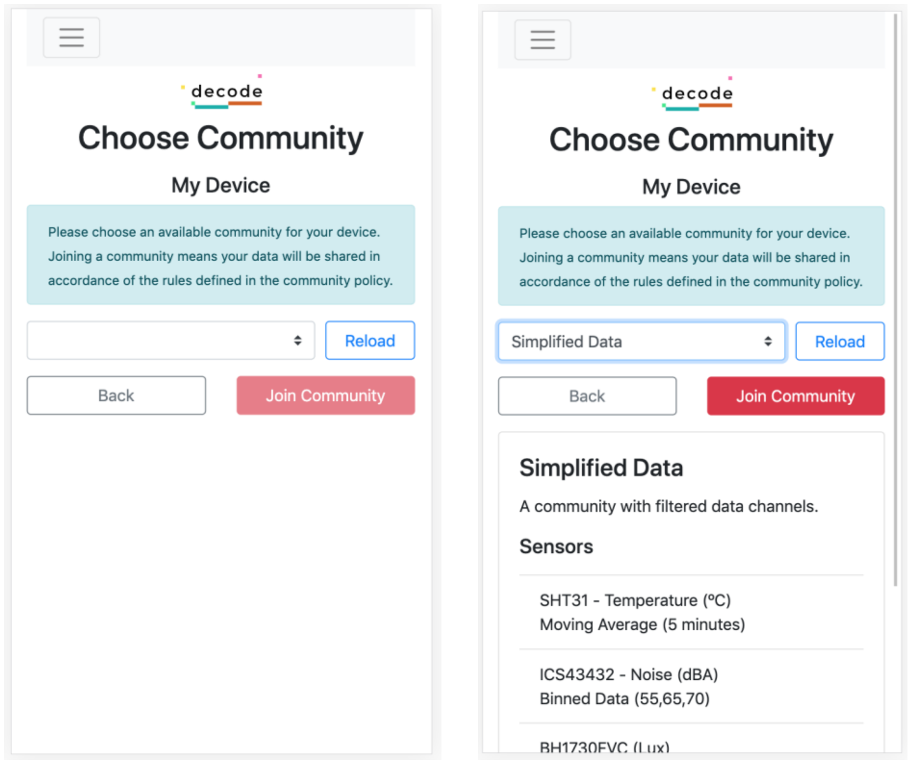
Are you sure you wish to add this device into the app? This is required if you wish to be able to control how its data is processed and transmitted

Figure 8 – DECODE webapp new device pages

If the user agrees the details are correct, they can press Confirm which opens a final confirmation popup window:

Once the user clicks through this page, they will have successfully added information about their device into the DECODE webapp, however at this point they have not joined any community, so no encrypted data will be being collected or stored.

To streamline the workflow for new users, the webapp now immediately redirects the user to a page which allows them to choose a community to join.


decode

Choose Community

My Device

Please choose an available community for your device. Joining a community means your data will be shared in accordance of the rules defined in the community policy.

decode

Choose Community

My Device

Please choose an available community for your device. Joining a community means your data will be shared in accordance of the rules defined in the community policy.

Simplified Data

A community with filtered data channels.

Sensors

SHT31 - Temperature (ºC)
Moving Average (5 minutes)

ICS43432 - Noise (dBA)
Binned Data (55,65,70)

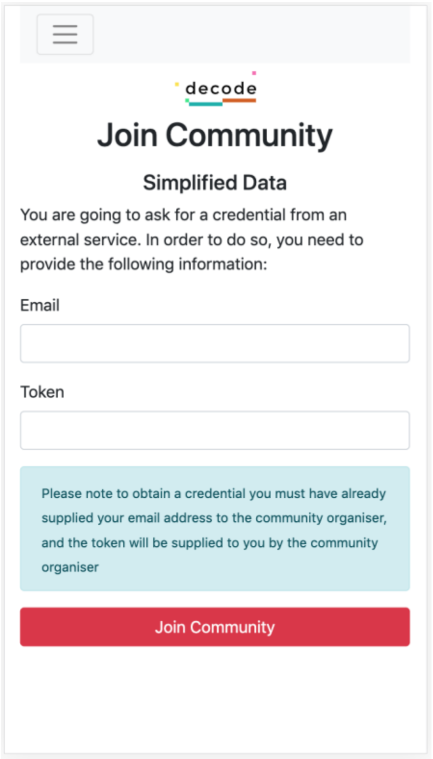
BH1730FVC (Lux)

Figure 9 - DECODE webapp choose community page

The user will then be prompted to choose a community from the drop-down menu list. To populate the drop-down menu the webapp makes a request to the Policystore service created by Thingful. As the user selects a community, we display details of the community policy operations on the screen so that the user can be an informed decision on how their data should be processed.

Once a user chooses a community, the DECODE webapp needs to do some of the complicated cryptographic operations in order to create a credential which allows them to prove to the dashboard that they have joined that community.

To do this the DECODE webapp prompts the user to enter some additional details which are required in order to obtain a credential. These details currently are an email address (which must be registered in advance), as well as a token value that is intended to be something that will be distributed by community organisers to participants.



Join Community

Simplified Data

You are going to ask for a credential from an external service. In order to do so, you need to provide the following information:

Email

Token

Please note to obtain a credential you must have already supplied your email address to the community organiser, and the token will be supplied to you by the community organiser

Figure 10 - DECODE webapp join community page

On submitting this email address and token the DECODE webapp will execute Zencode scripts using Zenroom via its JavaScript/WASM[1] bindings directly in the browser in order to create a credential request. This cryptographic material will then be sent to the Credential service provided by Dyne.org which checks the validity of the request, and if the supplied details are valid, will return a credential that the DECODE webapp must store. In addition, the webapp will also at this point make a call to the Encoder service developed by Thingful which creates the encrypted stream for the device which starts collecting data for that device, processing it and writing it to the encrypted data store.

The BarcelonaNow (BCNNow) tool has a data collector developed by Eurecat which is already configured to collect data for each community, so once the user has managed to make it through the process for creating a stream and adding their data into the community pool, their data will also start being collected the by the data collector.

If the user wants to then access this dashboard in order to be able to see their data in the context of other devices also part of the community, they can open the BCNNow tool URL, where they will see the screen shown in Figure 11 - BarcelonaNow login page.

[1] WASM stands for Web Assembly which is an emerging technology to develop a new standard binary runtime for the web browser environment. Please see https://webassembly.org/ for more details.

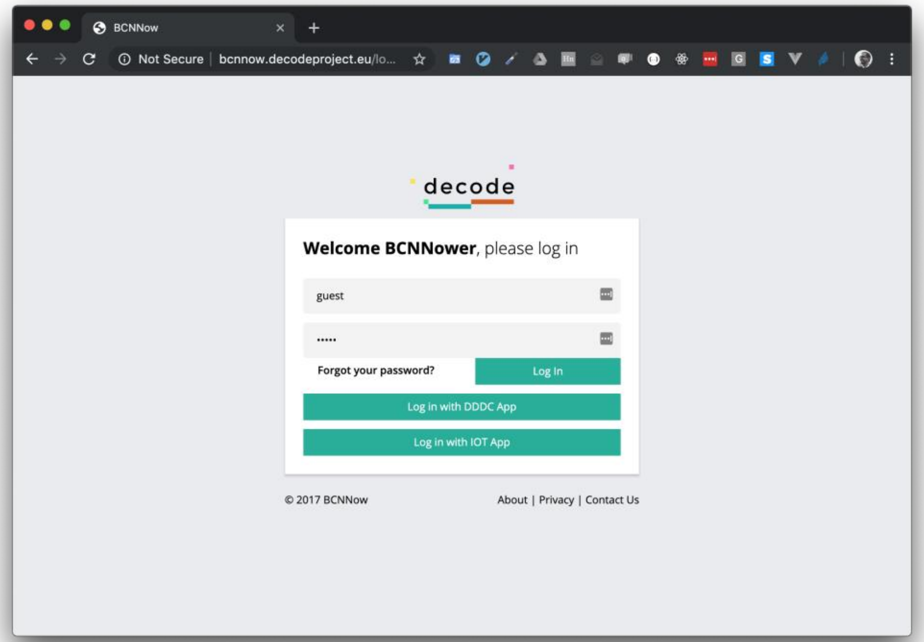

BCNNow
Not Secure | bcnnow.decodeproject.eu/lo...
decode
Welcome BCNNower, please log in
guest
Forgot your password?
Log In
Log in with DDDC App
Log in with IOT App
© 2017 BCNNow
About | Privacy | Contact Us

Figure 11 - BarcelonaNow login page

To login, the user must then click on the button labelled Log in with IOT App, which generates the QR code shown in Figure 12 - BarcelonaNow QR code page.

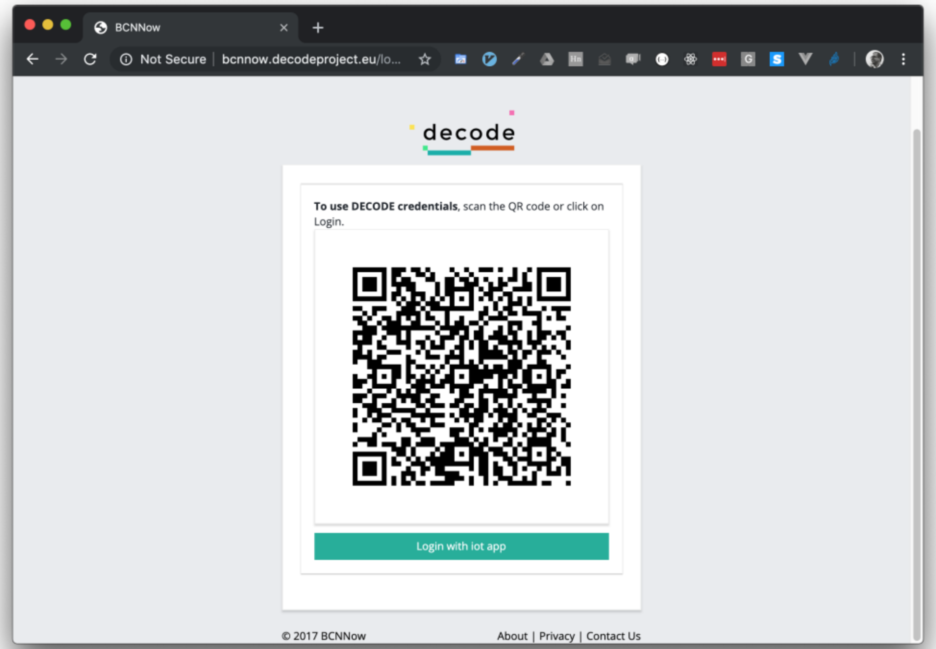

BCNNow
Not Secure | bcnnow.decodeproject.eu/lo...
decode
To use DECODE credentials, scan the QR code or click on Login.
Login with iot app
© 2017 BCNNow
About | Privacy | Contact Us

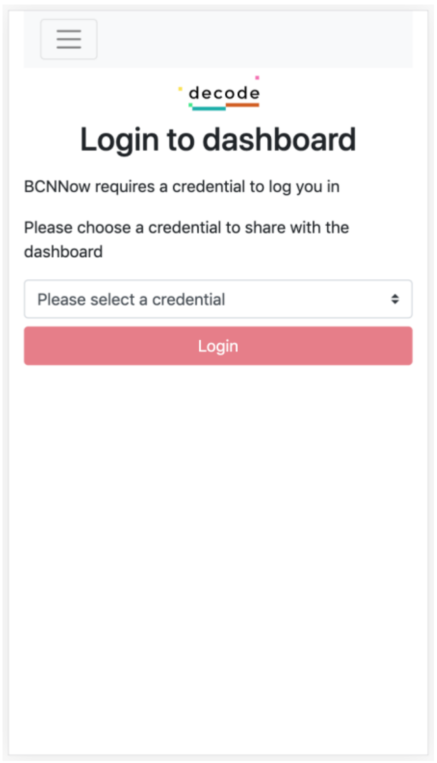
Figure 12 - BarcelonaNow QR code page

The user is prompted to scan the above QR code which contains the parameters defined in Table 2 - Parameters for BCNNow login URL.

Table 2 - Parameters for BCNNow login URL

Parameter	Description
`sessionId`	A unique session identifier created by the BCNNow tool which allows the system to identify the user sending the request so that they can subsequently be automatically signed in.
`callback`	The URL to which the DECODE webapp must send back the blind proof credential in order to prove the user should have access to the resource

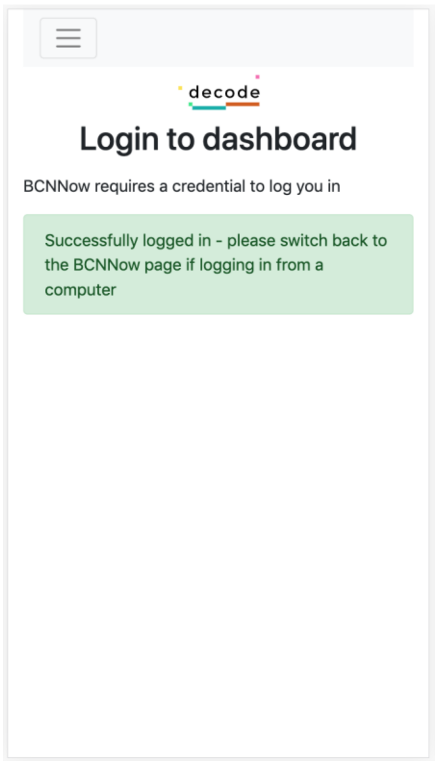
Once the DECODE webapp is opened with the above parameters, it opens a page which allows the user to choose whether to share their credential with the requesting service (shown in Figure 13 - DECODE webapp authentication pages).



Login to dashboard

BCNNow requires a credential to log you in

Please choose a credential to share with the dashboard



Login to dashboard

BCNNow requires a credential to log you in

Successfully logged in - please switch back to the BCNNow page if logging in from a computer

Figure 13 - DECODE webapp authentication pages

6.1.6 Software License

The application has been released under the terms of the GNU Affero General Public License as agreed for DECODE projects. Full details of the software license can be found here:

https://github.com/thingful/decodeweb/blob/master/LICENSE

6.1.7 Deployment

At time of writing the application has been deployed to a general web platform called Heroku (https://www.heroku.com). Heroku is a generalised platform for hosting web applications that is agnostic of any one server language or technology.

However, the application is deployed to Heroku within a Docker image, so is designed to be easy to move to a new hosting platform as and when that is required by consortium partners.

7 Conclusions

At the time of writing the DECODE webapp has been successfully deployed and has undergone preliminary testing with some beta testers at a UX session organised by IMI and Dribia in accordance with SmartCitizen. So far reports indicate positive results, and the SmartCitizen and consortium partners are moving ahead with a continuing series of workshops at which we hope to see a more significant deployment of devices allowing us to collect a meaningful dataset and explore the implications of this novel method of trying to give control back to end users in the field of IoT.

It remains a challenge to explain to users why these mechanisms are necessary; given the devices used for the IoT pilot are purely collecting ambient environmental data it can be hard to explain to users a) why they might want to apply strong encryption to this data, and b) what is the benefit of our attempt to put control of that data firmly in the hands of the users generating it. Future work on this might explore more personal IoT data, for example data collected from health-related devices, e.g. weight or blood pressure. It's much easier for users to understand the importance of protecting this sort of data so it might be easier to gain acceptance by focussing on these areas.

8 Appendices

8.1 Coconut

Detailed discussion of Coconut [6] is beyond the scope of this report but this brief appendix aims to give a very high-level overview of what Coconut is and how we are using it in the DECODE IoT pilot. Coconut describes itself as a selective disclosure credential scheme supporting distributed threshold issuance, public and private attributes, re-randomization, and multiple unlinkable selective attribute revelations.

What we take from this is it allows us to build a system where a user can choose to join a community or communities and as part of the process of joining that community can create a credential that allows them to cryptographically prove that they have a valid credential for the community without revealing anything else about themselves. During the onboarding of devices, the user creates a blind proof credential which is a complex piece of cryptographic material which allows the intended recipient to prove that the holder does indeed hold the piece of information or attribute that is claimed, but no other recipient is able to determine that information from the encrypted packet.

8.2 Zenroom

Zenroom [16] has been mentioned in our previous deliverable but is worth mentioning again here. We are already using Zenroom for encrypting data within the Encoder service previously developed, however it also plays a crucial role in our usage of Coconut. The Coconut scheme is described in the above academic paper, but it describes an advanced and experimental scheme that is beyond the expertise of Thingful to implement. Dyne.org however were able to build a working implementation of Coconut into Zenroom meaning that we were able to integrate this scheme into our functional prototype system.

9 Bibliography

[1] "Thingful," [Online]. Available: https://www.thingful.net/. [Accessed 24th May 2019].

[2] S. Mulube, "D3.7 Control and entitlement system for sensor data owners," DECODE, H2020–ICT-2016-1, 2018.

[3] "DDDC Participatory Platform," [Online]. Available: https://dddc.decodeproject.eu/. [Accessed 24th May 2019].

[4] "Dribia," [Online]. Available: http://www.dribia.com/. [Accessed 24th May 2019].

[5] "Dyne.org," [Online]. Available: https://www.dyne.org/. [Accessed 24th May 2019].

[6] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn and G. Danezis, "Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers," 8th Aug 2018. [Online]. Available: https://arxiv.org/abs/1802.07344. [Accessed 24th May 2019].

[7] "Eurecat," [Online]. Available: https://eurecat.org/. [Accessed 24th May 2019].

[8] "L'Institut Municipal d'Informàtica," [Online]. Available: http://ajuntament.barcelona.cat/imi/. [Accessed 24th May 2019].

[9] "SmartCitizen," [Online]. Available: https://smartcitizen.me/. [Accessed 24th May 2019].

[10] "Zenroom - crypto language VM," Dyne.org, [Online]. Available: https://zenroom.dyne.org/. [Accessed 24th May 2019].

[11] "React Native," Facebook, [Online]. Available: https://facebook.github.io/react-native/.

[12] "The Go Programming Language," [Online]. Available: https://golang.org/. [Accessed 24th May 2019].

[13] "The Python Programming Language," [Online]. Available: https://www.python.org/. [Accessed 24th May 2019].

[14] O. Sagarra, J. Rodríguez and P. Balcells, "D5.6 Deployment of Pilots in Barcelona," 2018.

[15] "Window.localStorage," Mozilla MDN Web Docs, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage.

[16] Dyne.org, "Zenroom - crypto language VM," [Online]. Available: https://zenroom.dyne.org/. [Accessed 17th May 2019].

[17] "DECODE Mobile App," [Online]. Available: https://github.com/DECODEproject/decode-mobile-app.