OAuthing: Privacy-enhancing Federation for the Internet of Things

Paul Fremantle
School of Computing, University of Portsmouth, UK
Email: paul.fremantle@port.ac.uk

Benjamin Aziz
School of Computing, University of Portsmouth, UK
Email: benjamin.aziz@port.ac.uk

***Abstract*—The Internet of Things (IoT) has significant security and privacy risks. Currently, most devices connect to a cloud service that is provided by the manufacturer of the device.**

We outline a proposed model for IoT that allows the identity of users and devices to be federated. Users and devices are issued with secure, random, anonymised identities that are not shared with third-parties. We demonstrate how devices can be connected to third-party applications without inherently de-anonymising them. Sensor data and actuator commands are federated through APIs to cloud services. All access to device data and commands is based on explicit consent from users. Each user's data is handled by a personal cloud instance providing improved security and isolation.

We demonstrate this model is workable with a prototype system that implements the major features of the model. We present experiment results including performance, capacity and cost metrics from the prototype. We compare this work with other related work, and outline areas for discussion and future work.

I. Introduction

IoT devices are proliferating throughout the world, bringing with them a significant threat to privacy and security. There are multiple concerns with IoT devices. One of the key issues in the IoT space is the concern that when a device is purchased, it is tied to a specific web system or Cloud Service (CS), which is owned and managed by a Cloud Service Provider (CSP). Issues include: the CSP may not be trustworthy; the CSP may be hacked; or may go out of business rendering the device inoperable. A common attack on CSPs has been to steal credentials and publish users' passwords.

In addition, there are other security and privacy concerns. Small, inexpensive and/or low-power devices do not properly support encryption allowing communications to be stolen. Few devices use well-defined identity models, meaning that spoofing attacks are possible. Data may be validly shared by multiple cloud services but then aggregated to de-anonymise user information and infringe on privacy.

We outline a new model, together with a prototype and experimental results that aims to address these issues. We call this model *OAuthing*.

A. Contributions

The contributions of this work in addressing these issues include:

- An architecture and system model that allows the decoupling of multiple parties: the manufacturer, the identity provider, the device identity management, and cloud services and applications that require access to IoT data. This decoupling encourages choice of provider as well as reducing the data available in any given attack.
- Clearly defined device and user registration processes, based on the OAuth2 protocol, that have been extended to support IoT devices and be effective in device scenarios.
- A model for providing anonymous identities for users, reducing the chance that leaked data can be tied to users.
- An architecture that provides each user with a separate cloud instance to handle sharing device data, and an approach for dynamically provisioning these cloud instances.
- A demonstration of the workability of the model through the creation of a working prototype, and experimental results including performance, cost and capacity metrics of the prototype.

B. Outline of the paper

The remainder of this paper continues with the following organization. In section II we propose a model of the different parties involved and an architecture which supports decoupling of different actors in the environment. In section III we outline a prototype middleware system that we have created to implement this model, alongside prototype devices and cloud services that demonstrate end-to-end flows. In section IV we outline the experimental results, together with the test harness that was built to gather data. We discuss the related work and compare it to our results in section V.w In section VI we analyse the outcomes of the experimental results and the overall model. We also outline a set of further work identified during this research.

II. Model

A. Participants

Figure 1 shows the current situation for many IoT systems, where there is no federation and the device talks to a single service that manages identity, stores data, provides a user web interface, etc. By comparison, the federated model we present in Figure 2 allows different federated parties to provide different services that work together. We call this model *OAuthing*. Figure 3 shows the UML sequence diagram of a runtime interaction between a device and a cloud service.

The participants of the OAuthing model are:

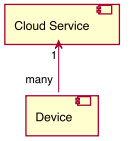

Fig. 1: Existing model

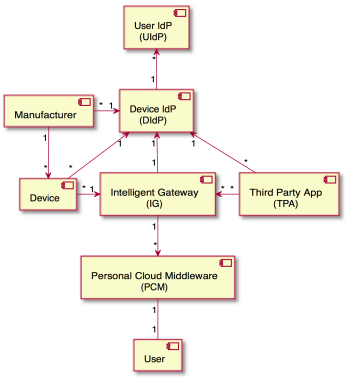

Fig. 2: Proposed Model

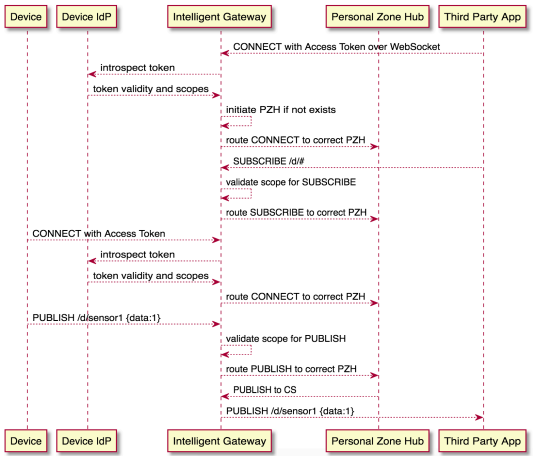

Fig. 3: Device publishing data to CS

- **The User Identity Provider (UIdP)**: this is an existing login system where Users present their credentials (e.g. Google, Facebook, Github, Twitter, or any OIDC login).
- **The User**: A User may own one or more Devices. A User must have at least one identity with a UIdP.
- **The Device Identity Provider (DIdP)**: this is an *Identity Broker* that first authenticates a User with a UIdP using existing federated identity protocols including OAuth2, OpenID Connect (OIDC), or SAML2. Once the identity is validated it then creates a secure random anonymous identity which is used in all further processing. This anonymous identity is not shared except with the Intelligent Gateway. Devices and Cloud Services are issued with random tokens that give permission to perform certain actions but do not identify users in any way. Currently, each instance of OAuthing has a single DIdP.
- **Personal Cloud Middleware (PCM)**: this is an isolated broker that shares data between devices and Third Party Applications (TPAs) on behalf of the user. The PCM talks to the Devices and the TPAs. Within the remit of a single OAuthing instance there is one PCM per user. We utilize a cloud environment to dynamically launch PCM instances on behalf of users as needed.
- **Intelligent Gateway (IG)**: The IG interfaces with the DIdP to validate identities and access authorization policies and to the cloud infrastructure to instantiate new PCMs. Devices and CSs connect to the IG, and it routes requests to each user's PCM.
- **Third Party Application (TPA)**: A device is an IoT device if and only if it shares or receives data and commands with an Internet service. Users control which TPAs can access their sensor data or control their actuators by explicitly consenting to authorize a TPA. Any third party can provide a TPA. If no TPA is authorized by the user then a Device's data is neither shared nor stored.
- **The Device**: The device consists of one or more sensors and actuators together with a controller. The device is issued with a Client ID at manufacturing time. Once the device is registered with a user, it stores a token that identifies the user, the Client ID and the scopes of access that the user has authorized.
- **The Manufacturer**: The Manufacturer is the logical organisation that creates and markets the Device, irrespective of whether they actually outsource any part of the physical manufacturing to a third party. In this model, the Manufacturer configures each device with a single DIdP.

The OAuth2 protocol [1] is a widely used federated authentication and authorization protocol. This model utilises the OAuth2 model as a basis for the identity and ownership of devices. One concern with IoT is that hardware devices can be compromised and secrets read from them. It is therefore important that each device has its own credentials. We map each device to be a unique OAuth2 Client, and we use the OAuth2 Client ID as a secure device ID that is only ever shared with the DIpD. We define ownership of a device by the user authorizing the issuance a security token to the device giving it permission to act on the user's behalf.

B. *Lifecycle*

We modelled the lifecycle of a device. The UML lifecycle diagram is shown in Figure 4.

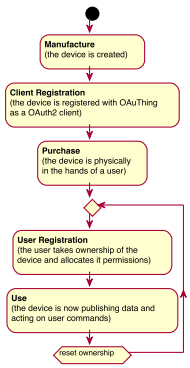

Fig. 4: Lifecycle of a Device

	UIdP	DIdP	Mfr	Dev	IG	TPA
User Profile	✔	✔				
HW ID			✔	✔		
Secure Device ID		✔	✔	✔		
Device Secret		✔		✔		
Pseudonym		✔			✔	
Token		✔		✔	✔	
Device Data				✔	✔	✔

Fig. 5: Information Visibility Matrix

Once the device is initially flashed it is connected to a manufacturing server. The manufacturer then uses the DCR API into the DIpD to request a Client ID and Secret. These are configured into the device by the manufacturing server. At the same time, the DIpD returns a unique User Registration URL (URU) that is printed onto the device (usually as a QR code) by the manufacturer.

When the user buys the device, they scan the QR code or otherwise access the URU. This directs the user to the DIpD which presents a choice of UIpDs to the user. Once the user is authorized with their existing UIpD, they are asked in turn to authorize the device. The resulting OAuth2 refresh token is then stored on the device, and represents the logical ownership of the device.

If at some future point the user sells the device they can revoke the OAuth2 token - either by resetting the device back to its initial state or by using a web interface at the DIdP.

C. Personal Cloud Middleware

A key part of the model is the concept of a personal hub: where each user's data is routed to its' own hub, protecting the data from multi-tenant attacks. Each hub is run in its own virtualised Cloud environment. When a request comes in from a device or CS, we use the anonymous identity associated with the bearer token to route the request to an instance that is specific to that user. If there is no cloud server available, the routing system makes a call to the cloud management system to instantiate a new PCM "on-demand", and then waits until the instance is running before routing the request to the PCM. In the model the PCM supports routing, distribution of data and commands, as well as summarisation and filtering of data. These capabilities have an important role in protecting users privacy: firstly, the runtime does not inherently share data such as IP addresses or MAC addresses that can be used to identify devices or users. Secondly, by filtering or summarising data, the PCM can avoid many fingerprinting attacks on devices [2]. The PCM can also provide protocol mapping and device shadow capabilities, meaning that it is simpler for TPAs to connect to devices.

D. Information visibility

The model allows us to analyse which identifiers and data each participant has access to. A key aim of this model is to ensure that each party has a restricted view of a users actions and therefore can only breach privacy in a limited way or with the help of a third party.

In Figure 5 we identify each participant and show what access they have to credentials, identities and data.

The manufacturer only knows the original device identity (e.g. MAC address) and the Client ID that was issued at manufacturing time. Unless the user chooses to share information with the manufacturer, then the manufacturer does not know the owner of the device and do not see any device data.

The DIdP knows the original user profile as provided by the UIdP. This is only used to ensure that two tokens issued to the same user will result in messages being routed to the same PCM. The DIdP is aware of the device presence (because the device must regularly refresh its tokens, and because the IG must introspect those tokens). However, the DIdP does not see any of the device data or commands. The DIdP is aware of which TPAs a user is subscribed to, but does not know which devices are interacting with which TPAs. In the event the DIdP is compromised, the attackers cannot steal passwords or any user data — only the link between the UIdP user identity and the anonymous identity.

The IG is given an anonymous random ID for the user, through an *Introspection API* on the DIpD which is only available to IGs. The IG does not store any data. If it is compromised, only data flowing through the system can be stolen, and this cannot inherently be tied to a user or any device identifier. Both the IG and the DIpD would need to be compromised to tie that data to a specific user.

The device and any TPAs are not aware of the user's identity - they only see a refresh or bearer token and do not have the authorization to call the Introspection API. Therefore a TPA cannot deduce the user's identity from the OAuthing system. However, a TPA may be aware of the user's identity through out-of-band means. For example, if the user decides to share their IoT data publicly on a webpage and this webpage identifies them. Even in this case, the TPA does not know the device id, only the data that is shared. Alternatively, the data itself may contain personally identifiable traits: this may be mitigated by sharing a summary or filtered data. Finally, the device data itself is only visible to the device, the PCM and authorized TPAs.

III. Implementation

In order to validate the model, we built a prototype of the system. We did this to answer the following research questions: Is the proposed model implementable? Do any specific issues emerge during implementation? Does the implementation demonstrate *workability*: that this works in a cost-effective approach that could be used in practice? What performance does the prototype provide - in terms of memory usage, latency, transaction rates and cost per transaction? How does this performance compare to existing systems?

A. Protocol Mapping

The model has been designed to work independently of specific protocols. However, in order to implement a prototype we needed to make specific choices on protocols. We also provide a mapping of part of the OAuth2 protocol into MQTT [3] (an IoT-optimised protocol), allowing the device to utilize a single protocol for identity, authorization and data. This simplifies the device coding and reduces the memory footprint of OAuthing support.

To support interoperability of devices and CSs we specified a minimalist API for communicating data and commands, based on MQTT. This could be extended in future, with protocols such as HTTP and CoAP [4]. In our previous work, there was an authorization policy model that permitted complex rules. While this was more powerful, it was not easy

to present users with a clear consent choice. In this model, there is a simplified approach that is based on a simple *topic* model:

- `/d/#`: any messages sent to the `/d` topic or subtopics thereof are considered *data* published by sensor devices.
- `/c/#`: any messages sent to the `/c` topic or subtopics therof are considered *commands* published to device actuators.

Data should be published in the SenML [5] format, but other formats are allowed. Users can choose to authorize devices and CSs to read or write to these topics.

The following sections outline the components that were implemented to demonstrate the system.

B. The OAuthing DIdP

The prototype DIdP is called the OAuthing DIdP. We implemented a number of capabilities:

- The identity broker pattern: we allow users to utilize their existing identities with UIdPs and broker new OAuth2 tokens based on the tokens passed from third-party systems;
- an implementation of the Dynamic Client Registration API [6];
- an implementation of the OAuth2 Introspection API [7];
- a Cassandra persistence layer; and
- an *embedded* MQTT broker that supports the mapping of the OAuth2 Token API into the MQTT protocol.

We implemented the OAuthing DIdP as a set of containers running in the Docker[1] container system, allowing it to be efficiently deployed and tested in a cloud environment. In our system we do not store the users profile information. Instead we issue each user with a new secure random anonymous identifier. In order to associate this identity to the UIdP's identity, we take the name of the UIdP and the UIdPs unique identifier and hash these together and store this against the secure random identifier.

The embedded broker pattern solved a number of security issues that were present in earlier iterations of this work, including securing the OAuth2 Refresh flow.

C. IGNITE

The prototype of the IG is called IGNITE (Intelligent Gateway for Network IoT Environments). IGNITE runs in a Docker cloud container environment and has access to control this Docker environment. When a device or TPA initiates an MQTT connection with the `CONNECT` packet, IGNITE first validates the Bearer Token by calling the DIdP's introspection API. This either returns a random anonymous ID together with a set of scopes; or informs IGNITE that the token is invalid. IGNITE is then responsible for launching a new cloud instance to act as the PCM on behalf of the user, or routing the request to the existing PCM. IGNITE also implements the security policies defined in scopes by the DIpD, enforcing them before routing requests to the PCM.

D. Personal Cloud Middleware

The PCM was implemented using the open source RSMB MQTT broker[2]. This broker has a very low memory overhead, and enabled us to run a significant number of PCM containers on standard hardware. We have not yet implemented summarisation and filtering on the PCM, which will potentially enlarge the memory footprint, but we did not yet optimise the Docker runtime of the PCMs or the underlying Operating System, and therefore we are confident that this can be offset.

E. Device Hardware

We aimed to build the simplest possible device to provide a baseline evaluation of whether the model was implementable on very small footprint devices. We chose the commonly available ESP8266 platform for our reference device. This chip provides an embedded 32-bit processor, Wifi connectivity and a number of digital inputs and outputs for less than US$2.50 each (at the time of writing). ESP8266 supports TLS without full certificate authority chains. Instead, it uses fingerprints of SSL certificates to validate them. The device is configured with the DIdP fingerprint at manufacturing time and then the IG fingerprint at User Registration time (using the DIpD as the trusted source for the IG fingerprint).

F. Manufacturing Process

Once the device is operational, it contacts a local manufacturing server via MQTT. The server may also have a physical means of identifying the device (e.g. scanning a bar code on the device). The manufacturing server calls the DCR API on the DIdP to register the device, and then sends the Client ID and Secret to the device over MQTT, where they are stored in EEPROM. The QR Code is then printed and attached to the device. When this QR code is scanned, the DIdP first checks that the device is switched on and connected to the DIdP. It then prompts the user to login via a UIdP (e.g. redirecting to Google's authentication). Once the user is logged on the user is asked to authorize the device. This initiates a modified OAuth2 flow with the device, resulting in the refresh token being stored in the device's EEPROM. The device is now ready for use.

G. Sample Third Party Application

In order to demonstrate this system we also created a simple web-based cloud service. This first connects to the OAuthing DIdP using a standard OAuth2 HTTP flow to request access to IGNITE. The user logs in using the same UIdP that they registered their device with. After the user authorizes the Sample TPA to access IGNITE the sample app is loaded. This uses MQTT over WebSockets to communicate, and presents a simple UI allowing users to interact with the device.

IV. Evaluation

The main result is that there is a working prototype of the model that demonstrates all aspects of this decoupled

[1] http://www.docker.com

[2] https://github.com/eclipse/mosquitto.rsmb

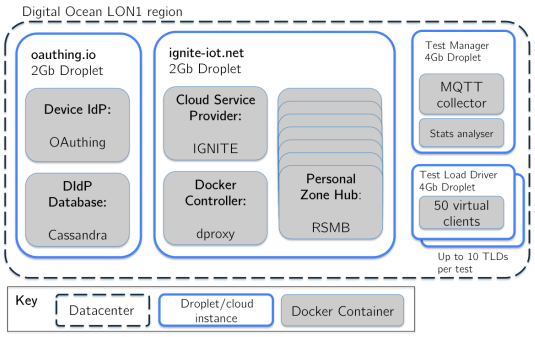

Fig. 6: Test Environment

approach, including support on a low-cost device and a Cloud implementation of the server-side components[3].

A. Results

Figure 6 shows a simplified diagram of the test environment, which is running in a public cloud environment.

Figure 7 shows the latency comparison of IGNITE compared to Mosquitto (a standard MQTT server) using the *one second client*. Figure 8 shows the mean, 95% and 99% percentiles for the IGNITE latency responses in the same test. The graph demonstrates that IGNITE shows consistently low latencies across all workloads. The additional latency added to message interactions compared to Mosquitto was around 1ms. The percentiles show that 99% of requests had latency of less than 11ms even when the system was loaded with 400 test clients, and 95% of requests had latency less than 6ms. Given that average round trip ping times over the Internet are in the 20-80ms range, these are good results.

The next data point collected was the maximum number of PCMs that could be run on a single cloud server. The tests were run on a server with 2Gb of memory and no swap configured, costing US$20/month. This environment was able support at least 400 PCM instances, with the server running out of memory beyond 415 containers. Simply adding swap will increase this number at the cost of some latency, but we have not yet evaluated this balance.

Figure 9 shows the average connect time for three different scenarios. The fastest is the Mosquitto broker, with an average connect time of 24.5ms. The slowest is the IGNITE when the user has not previously connected. In this scenario, the system needs to introspect the token and then wait until the new container is launched and ready. This takes on average 1294ms (1.3 seconds). While this is comparatively large, it happens rarely in the system and in practice devices take between 2 and 10 seconds to connect to local Wifi networks. It could be ameliorated by pre-loading unused containers and associating them with users at connect time.

The third scenario is the connect time when there is already a user container running. The average time here was 35.9ms. The extra latency compared with Mosquitto (35.9ms vs 24.5ms) is well within acceptable ranges.

[3]A short video showing the registration and data sharing process is available at http://freo.me/oauthing-video

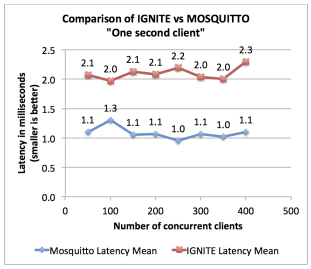

Fig. 7: One second client: IGNITE vs Mosquitto

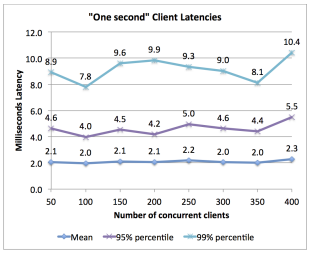

Fig. 8: Latency percentiles for IGNITE

Finally, figure 10 shows the performance of the IGNITE system under stress. This shows that the server was handling more than 4500 sent messages per second at all levels and the average latency rose to 83.3ms when the system had 400 concurrent clients. This test shows that the system would support each user owning 600 devices each interacting once a minute, even when the system is fully loaded with 400 concurrent PCM containers. The latency line shows that as new clients are added the latency increases in direct proportion, demonstrating fair allocation of resources.

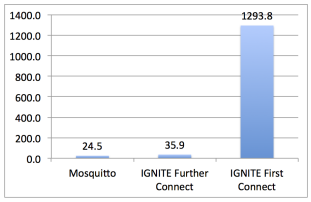

Fig. 9: MQTT Connect Time

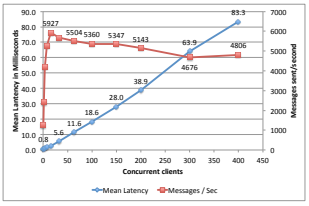

Fig. 10: Stress Client Latency and Throughput

V. Related Work

In [8] we previously looked at using Federated Identities for IoT, especially mapping OAuth2 tokens to work with the MQTT protocol. In a follow up work [9], we demonstrated the use of the *Dynamic Client Registration* (DCR) API to support each device having a unique OAuth2 client identifier. In each case identifiers needed to be manually added to the device, which is unrealistic in manufacturing processes. Existing public IoT middleware such as IBM Watson IoT and AWS IoT also have this concern. Compared to these previous works, this current model adds a clear automated device and user registration process. It also adds anonymous identities and Personal Cloud Middleware.

IOT-OAS [10] addresses the use of OAuth2 with the CoAP protocol. The mapping of the OAuth2 Token API to support IoT devices using the CoAP protocol is being formalised in [11], and is described in [12]. In [13] there is a demonstration of the OAuth1 protocol with MQTT, favouring OAuth1 over OAuth2 for IoT devices. The reasons for choosing the older OAuth protocol are obviated by the mapping of the

refresh flow which OAuthing offers. In [14] and in [15] there are platforms that support OAuth2 for IoT devices that communicate via HTTP and WebSockets. None of these works address automated registration processes, and none provide the privacy controls of anonymous identifiers or isolated personal cloud instances.

In [16] a capability-based access system is described that allows anonymous identities to be used. [17] provides an Architecture Reference Model for an approach that supports anonymous identities. Neither of these systems separate the provision of anonymous identities from the data-sharing middleware.

The concept of a Personal Zone Hub (PZH) is described in the Webinos [18] system: this is similar to our PCM. However, in Webinos users must instantiate the PZH themselves, and there is no analysis of the cost per user. We extend the PZH concept to support a dynamic instantiation of PCMs as containers and provide a cost model. Webinos does not address secure federated device identities and does not provide a registration process.

VI. Discussion and conclusions

The OAuthing model provides significant improvements over existing systems, providing much stronger guarantees of privacy. Data and identity are not shared without consent, and data can be shared anonymously. Device and user registration are automated, and the PCM model can prevent fingerprinting, sharing of IP and MAC addresses, as well as user and device identity.

A key concern around the PCM model is that the cost per user might be too high. The prototype demonstrates that PCMs can be automatically deployed on behalf of the user with acceptable times. The experimental results demonstrate that a US$20/month cloud server can support 400 users, resulting in a cost per user of just $0.05 per month. Further optimization could reduce this cost.

The OAuthing model and prototype demonstrate that devices can be connected to TPAs without inherently leaking the user's identity to either system. User's may choose to provide TPAs with their identity, but that becomes a positive consent of the user rather than the default. In addition users can bring pre-existing identities to the system rather than being required to create new credentials, which reduces the chances of password theft and gives users a choice of identity provider.

There remain a number of unexplored aspects of this model. We expect to add tests that evaluate the performance of the OAuthing DIdP. In addition we plan to extend the system to support multiple co-existing DIdPs. We have discussed this model with two significant device manufacturers. One potential concern is that some device manufacturers' business models are based around collecting user data and therefore this system is unattractive precisely because it improves users privacy. However, there are a number of areas where this approach offers significant benefits, for example in medical devices, where the manufacturer does not wish to access data for regulatory reasons. In addition, it is possible to start with the OAuthing model and progressively lessen certain privacy controls to provide a system that still shares data but provides stronger guarantees.

We have identified a number of future items of research around this, including developing a full threat model for the system; supporting devices that communicate via gateways (e.g. Bluetooth devices talking to a phone or hub); and demonstrating clustering and high-availability for the OAuthing DIdP and IGNITE. We have also identified that further de-centralisation maybe possible by utilizing distributed ledger technologies such as Blockchains with the DIdP to reduce the requirement to have a central IdP for devices.

References

[1] D. Hammer-Lahav and D. Hardt, "The oauth2.0 authorization protocol. 2011," IETF Internet Draft, Tech. Rep., 2011.

[2] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[3] D. Locke, "MQ Telemetry Transport (MQTT) V3.1 Protocol Specification," 2010.

[4] Z. Shelby, K. Hartke, and C. Bormann, "Constrained Application Protocol (CoAP) draft-ietf-core-coap-18," https://tools.ietf.org/html/draft-ietf-core-coap-18, accessed: 2015-03-13.

[5] J. et al, "draft-jennings-core-senml-06 - media types for sensor markup language (senml)," https://tools.ietf.org/html/draft-jennings-core-senml-06, (Accessed on 30th August 2016).

[6] J. Richer, M. Jones, J. Bradley, and M. Machulak, "Oauth 2.0 dynamic client registration protocol," Tech. Rep., 2015.

[7] J. Richer, "Oauth 2.0 token introspection," Tech. Rep., 2015.

[8] P. Fremantle, B. Aziz, P. Scott, and J. Kopecky, "Federated Identity and Access Management for the Internet of Things," in *3rd International Workshop on the Secure IoT*, 2014.

[9] P. Fremantle, J. Kopeckỳ, and B. Aziz, "Web api management meets the internet of things," in *Services and Applications over Linked APIs and Data SALAD2015*, 2015.

[10] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: An OAuth-based Authorization Service Architecture for Secure Services in IoT Scenarios," 2015.

[11] IETF, "Authentication and authorization for constrained environments (ace) - documents," https://datatracker.ietf.org/wg/ace/documents/, (Accessed on 30th August 2016).

[12] H. Tschofenig, "Fixing user authentication for the internet of things (iot)," *Datenschutz und Datensicherheit-DuD*, vol. 40, no. 4, pp. 222–224, 2016.

[13] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, and A. Panya, "Authorization mechanism for mqtt-based internet of things," in *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016, pp. 290–295.

[14] D. Raggett, "Compose: An open source cloud-based scalable iot services platform," *ERCIM News*, vol. 101, pp. 30–31, 2015.

[15] S. Emerson, Y.-K. Choi, D.-Y. Hwang, K.-S. Kim, and K.-H. Kim, "An oauth based authentication mechanism for iot networks," in *Information and Communication Technology Convergence (ICTC), 2015 International Conference on*. IEEE, 2015, pp. 1072–1074.

[16] D. Rotondi, C. Seccia, and S. Piccione, "Access control & iot: Capability based authorization access control system," in *1st IoT International Forum, Berlin, November*, 2011.

[17] J. B. Bernabe, J. L. Hernández, M. V. Moreno, and A. F. S. Gomez, "Privacy-preserving security framework for a social-aware internet of things," in *International Conference on Ubiquitous Computing and Ambient Intelligence*. Springer, 2014, pp. 408–415.

[18] H. Desruelle, J. Lyle, S. Isenberg, and F. Gielen, "On the challenges of building a web-based ubiquitous application platform," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 733–736.