Adaptive Context Environments (ACE) Lab

2018

Deriving privacy and security considerations for CORE

Morris, Alexis and Lessio, Nadine

Deriving Privacy and Security Considerations for CORE: An Indoor IoT Adaptive Context Environment

Alexis Morris*
OCAD University
Toronto, Ontario
amorris@faculty.ocadu.ca

Nadine Lessio
OCAD University
Toronto, Ontario
nlessio@faculty.ocadu.ca

ABSTRACT

The internet-of-things (IoT) consists of embedded devices and their networks of communication as they form decentralized frameworks of ubiquitous computing services. Within such decentralized systems the potential for malicious actors to impact the system is significant, with far-reaching consequences. Hence this work addresses the challenge of providing IoT systems engineers with a framework to elicit privacy and security design considerations, specifically for indoor adaptive smart environments. It introduces a new ambient intelligence indoor adaptive environment framework (CORE) which leverages multiple forms of data, and aims to elicit the privacy and security needs of this representative system. This contributes both a new adaptive IoT framework, but also an approach to systematically derive privacy and security design requirements via a combined and modified OCTAVE-Allegro and Privacy-by-Design methodology. This process also informs the future developments and evaluations of the CORE system, toward engineering more secure and private IoT systems.

CCS CONCEPTS

• **Human-centered computing** → **Mixed / augmented reality**; **Ambient intelligence**; **Ubiquitous and mobile computing design and evaluation methods**; *Contextual design*; • **Security and privacy** → *Mobile platform security*; *Distributed systems security*; *Mobile and wireless security*; *Domain-specific security and privacy architectures*;

KEYWORDS

Privacy; security; internet-of-things; architectural framework; ambient intelligence

ACM Reference Format:
Alexis Morris and Nadine Lessio. 2018. Deriving Privacy and Security Considerations for, CORE: An Indoor IoT Adaptive Context Environment. In *2nd International Workshop on Multimedia Privacy and Security (MPS '18), October 15, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3267357.3267363

*Corresponding author.

MPS '18, October 15, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5988-7/18/10…$15.00
https://doi.org/10.1145/3267357.3267363

1 INTRODUCTION

Ambient intelligent and context-aware environments describe technologies which have the property of being embedded, context-aware, personalized, adaptive, and anticipatory [1]. This fits the category of internet of things applications closely, wherein environments must make sense of user situations to inform reactive decision-making wherein the environment provides multiple (agent) behaviors on behalf of users. Context, in this sense, is important and typically defined as "any information that can be used to ascertain the situation of a persona or entity," [2, 19] while context awareness here refers to a systems capability for applying and making sense of context sources.

Context, however, is inherently multidimensional, encompassing a wide variety of signals and data sources (potentially for multiple users and scenarios). This demands that ambient intelligent systems designs are sensitive to changes in a host of potential and multidimensional contexts, for multiple users and scenarios. The internet of things (IoT) is a central mechanism for bringing these contexts to the forefront, allowing for the aggregation and sense-making that is needed to develop a common operating picture for an environment, toward situational awareness and decision-making for many IoT-specific applications and stakeholders, i.e., system users, service providers, and other actors or agents.

This often represents a distributed system, as a service-oriented architecture, and as a result of its distributed design introduces a host of security and privacy challenges. Security, here considered as maintaining the operation of the system as designed, without external influence of unauthorized actors – malicious or otherwise – is also multi-dimensional and varied. As such, the secure operation of each module within the internet of things application must be considered, alongside the many potential events impacting its operation. This includes security of decision-making and information processing and storage elements generating and maintaining information assets.

Privacy within the IoT can be considered in terms of security of information access within such systems, maintaining communication channels and the data being transferred across these channels. As a distributed system highly dependent on communication of potentially sensitive sensory information across complex and dynamic networks, the internet of things has high privacy needs that must be brought to the forefront, throughout the lifecycle of such systems (including design, development, deployment and activation, runtime, and deactivation). Hence, developing a functional and privacy-enhanced system is a considerable challenge in this complex systems landscape.

This work considers the privacy and security needs of adaptive and context aware environments, particularly for indoor uses, as a

step toward understanding these design considerations for internet of things deployments. This involves the design of a prototype system, based on a new CORE (COntextual REality) internet of things framework which has a multimedia focus, aimed at both the acquisition of context from within an indoor environment as well as a streamlined presentation of IoT system decisions made using this context. This represents an initial testbed design, with multiple data sources to be considered, specifically as it aims to apply advances in computer vision alongside traditional sensing mechanisms, and angles toward advanced visualization, particularly through mixed reality approaches. This system itself could also potentially have future applications in both outdoor and vehicular situations.

While the scope of this work is primarily to explore privacy and security considerations for an initial prototype of this CORE system, it also highlights the need to consider both "humans-in-the-loop" as well as "things-in-the-loop." In doing so, the target contributions of this work involve i) the presentation of a contextual architecture for internet of things systems, ii) an early prototype framework deployment, iii) an exploration of the derivation of security and privacy considerations for this framework. This targets future system developers and researchers of highly contextual internet of things systems. To discover the required design considerations, this stage of the work merges two existing frameworks, namely a minimized version of the OCTAVE-Allegro methodology for eliciting security considerations [9], alongside the Privacy-by-Design (PbD) principles of [18], toward systematically eliciting privacy and security in IoT systems engineering.

The process of deriving privacy ad security considerations is presented as follows: Section 1 has provided an overview of the need for both privacy and security in the internet of things domain, as a multi-dimensional problem. Section 2 briefly introduces the background to internet of things research, toward the CORE architecture, and elements of adaptive environment systems. Section 3 describes the CORE architecture from a multidimensional design perspective. Section 4 presents an initial indoor smart environment prototype based on the CORE architecture, and its design modules. Section 5 discusses the approach toward deriving security and privacy considerations for the CORE prototype, as implemented and highlights concepts of the OCTAVE-Allegro methodology and the Privacy-by-Design framework that have been applied. Section 6 provides a summary discussion of these approaches, and Section 7 concludes the paper.

2 INTERNET-OF-THINGS FOR ADAPTIVE CONTEXT ENVIRONMENTS

The Internet of Things (IoT) paradigm represents a wide range of interconnected networks of sensors, devices, and the infrastructures they enable and support. In this rapidly advancing technology, as seen by surveys like [7, 8], edge and cloud computation allows for sensing and sharing of data and knowledge across these systems, and their use cases range among a plethora of domains, whether for interconnected vehicles, or for interconnected individuals and homes. In this space a host of rich problems arise and are being addressed in terms of hardware and software, protocols, and usage parameters [16]. However, there remains a need for human-centered design approaches, as IoT interaction, enabled by mobile devices,

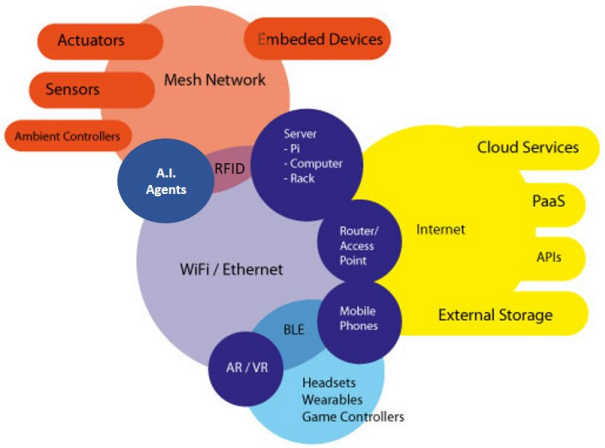

Figure 1: The technologies needed for adaptive context environments.

augmented reality head mounted displays, and other wearables and networked devices, is set to revolutionize society.

When considering an adaptive context environment, enabled by the internet of things, this involves: i) networks, ii) interconnectivity, iii) domain application hardware and software, and iv) internet online services. These are depicted in Figure 1, showing several overlapping elements. In terms of networks, this incorporates dynamic interactions between devices, embedded in the environment, which provide sensors, actuators, and ambient controllers. The interconnectivity of these core elements leverages telecommunication protocols, typically wirelessly (or directly via Ethernet) and provides for complex systems of communication. Within this, hardware and software technologies rely on the networking standards and protocols (such as HTTP, or TCP, etc). Examples of this include backend servers, routers and gateways, mobile devices, and even technologies like embedded processors. A number of domain applications also leverage the interconnectivity and sensor aspects of the internet of things, particularly for headsets, wearable technologies, and gaming, including virtual and augmented reality devices. Together with online services, providing access to platforms and API's, adaptive environments effects a large scale decentralization of the internet of things, including decentralized processing and storage services; this must be developed with an eye for the human factor issues and socio-technical systems challenges, like privacy and security, that everyday interaction and scenarios bring to the forefront, [24].

3 CORE ARCHITECTURE

As indicated, the aim of this work is to present a CORE IoT architectural framework for adaptive environments and to explore its security and privacy implications and considerations on a prototype. Specifically, the CORE (Contextual Reality) approach to smart environments engages a combination of perspectives, aimed at the dual system design targets of i) providing environmental context to

the IoT system controllers, and ii) providing appropriate system visualizations and interfaces for humans-in-the-loop of the IoT. This centers on six development perspectives, namely i) the virtual, ii) ambient, iii) collaborative, iv) informational context, v) inferential, and vi) networking perspectives. These are shown in Figure 2, and described as follows:

The *virtual* perspective targets the design of visual interfaces for humans-in-the-loop, with a focus toward next generation mixed augmented reality interfaces. In this, established computer graphics scenes are considers, such as using Unity3D, and controllers within Unity, like the ML-Agents framework. The ambient perspective aims toward the traditional designs of the internet of things, involving objects, object controllers, and control logic. This speaks to the awareness of actual environment details. This includes the presence of individuals within the system, and any interface hardware available, with AR viewers being highlighted.

The *collaborative* perspective highlights the notion that such systems must account for the fact that multiple users can engage within the environment, each having potentially unique goals, roles, states, and tasks. Additionally, multiple IoT objects, like cameras, microphones, speakers, and switches, are within the environment and may be used in service of these varied users.

Likewise, the *informational* context perspective addresses the need to consider the variety of information sources, and how those sources are provided to controllers for processing and decision-making. This may involve varied IoT objects and IoT agents as controllers and monitors of such objects.

The *inferential* perspective considers the potential for advances in machine learning and computer vision for making sense of situations through video feeds and other incoming information streams. In this architecture, trained data is used to provide models as an inference library, for instance using TensorFlow [35], for information agents within the system to access context from pattern recognition.

The *networking* perspective addresses the needs for communication within the system, particularly as middleware for the objects and agents within the system, on the one hand, and the various platforms in the cloud, on the other. Further, the access to information resources like the model library for inferencing are also considered. This networking perspective considers technologies like backend server frameworks like Redis [30] or Flask [11], protocol technologies like MQTT [25], and even socket and tunnel layers, like NGrok [26].

The next section deepens this architecture into a framework and early prototype, which centers heavily on the *ambient, inferential, informational, and networking perspectives* as a starting point for future research. This prototype will be considered in further sections from a security and privacy standpoint.

4 CORE PROTOTYPE

4.1 Overview

The CORE prototype implementation is shown in Figure 3, and represents initial components for an instantiation of the IoT aspects of the CORE architecture, with the visualization perspective left for future development. The prototype consists of two consumer voice interface devices (Google Home and Amazon Alexa [5, 15]), and a Pixel Phone running Google Assistant [13], each being able to use a shared camera sensor on a Raspberry Pi [29] to take pictures of objects, or faces, to identify them using two machine learning models. These models are served up as their own endpoints on a local-area-network(LAN), which are requested via webhooks on a Raspberry Pi device. Any Google Assistant enabled device or Alexa device can access this flow. This is a proof of concept scenario, which will inform future prototypes and examples.

The driving idea is to serve model processes as an API endpoint for requests by decentralized components. In this way, local area devices, regardless of their make or type, could simply send an HTTP request for batch processing, or singular processing of images and video frames. This was informed by Guinard's idea around a web of things [17] which looks at using web based architectures and protocols, to help manage the IoT in a more accessible and friendly way. It also gives the network some flexibility, as heavy processing can be offloaded from smaller devices to a larger server. The structure of using an API endpoint means that the model processes are available across network, rather than being confined to a singular device. This also gives the ability for the network to run model processes without the presence of the internet. Some devices may need to access cloud based services to operate, but others do not, and hence "things" within a sensor network, and any machine learning (ML) processes can operate normally within a local distributed network.

4.2 CORE Prototype Components

"There are three fundamental components that combine to form an IoT node: intelligence, sensing, and wireless communications" [39]. The CORE prototype addresses this notion via the elements highlighted below.

ML Models: This component highlights machine learning models, pre-trained on specific datasets, that are offered up as services to other devices on the network. They reside as their own contained processes on a backend server, and can handle classifications, such as identifying properties within images or video frames that are sent to them over the network. These are run on a Tensorflow framework with Keras API's for development [20]. Example pre-trained models would include object detection models like Tiny Yolo, Tiny-SSD [31, 38], or those trained on image and face-detection datasets like ImageNet, COCO, or FaceNet [21, 23, 33], or pose and gesture detection like Pose-net, [12, 27].

Backend Server: This component addresses the need for a backend server machine that runs locally on the network and handles the ML processes. In this instance, a combination of Redis, and Apache [6], serves up Flask endpoints for devices to access the ML processes, as in [20].

Embedded CORE Service Devices (ECSD): This component addresses service devices, including custom sensors, actuators and more general devices like Raspberry Pis. These differ from commercial service devices, and are arranged to work together as a sensor network. In this instance, this is managed by the Raspberry Pi device as a resource broker. These devices can include things like temperature or humidity sensors, light sensors, proximity devices, and room based cameras. Combined, these operate to gather the state of the room, and can be accessed by software or agents operating on the local-area-network (LAN) of the institution.

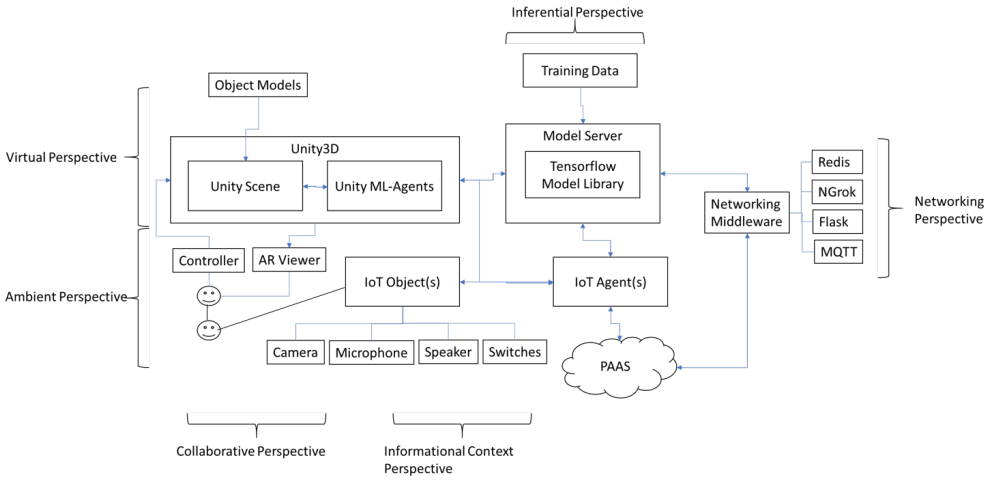

Figure 2: The CORE IoT framework for adaptive context aware internet of things, with both IoT and mixed reality visualisation elements.

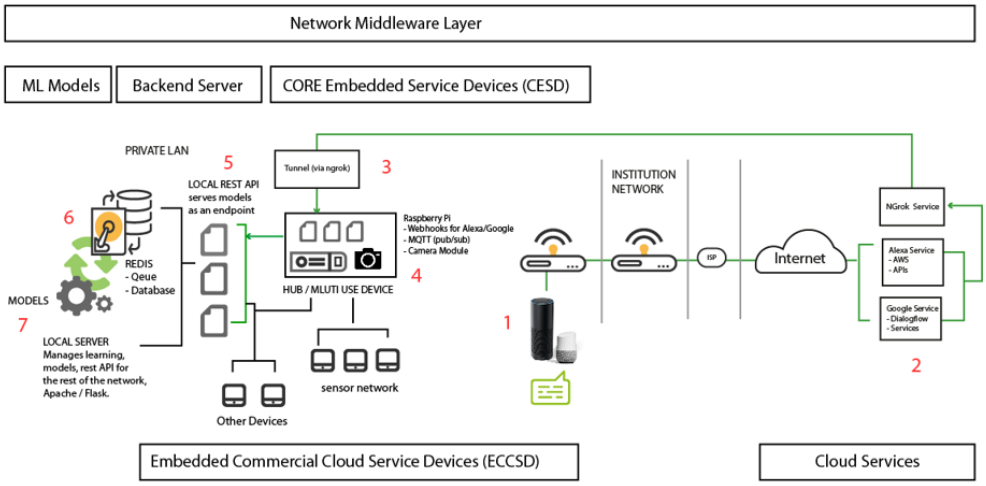

Figure 3: Early CORE prototoype set of components for implementing the IoT element of the architecture, leveraging [20].

Embedded Commercial Cloud Devices (ECCD): This component outlines devices within the system that require the support of a cloud based service to operate, or also for setup, initialization, and registration. They include items such as a Google Pixel Phone, Wemo Switches, Google Home, and Amazon Alexa assistant devices [5, 15, 37] – each are generally commercial devices that are tied closely to their parent company systems.

Cloud Services: This component addresses cloud services, i.e., remote services made available on demand via the internet. In this case this includes platforms such as Google Cloud [14] that provide services for Google Home, and natural language processing. Alexa Skills Kit for the Alexa devices [4], and Ngrok which provides a way to make secure tunnels to localhost. NGrok provides a way to host a webhook fulfillment, and to provide remote access for monitoring.

Network Middleware: This component addresses the middleware protocols and networking devices employed in system communication. This includes institutional routers (and portable travel routers), messaging protocols like MQTT [17], and web protocols like HTTP. the purpose of these protocols is to provide devices with a framework to communicate with one another both locally and remotely. This element of the system is of high significance.

4.3 Networking Designs and IoT Functionality

The network itself is broken down into ML Models, the backend server, CORE Embedded Service Devices, Embedded Commercial

Cloud Service Devices, Network Middleware, and Cloud Services. The models, API endpoints for the models, and the message broker that they communicate with, reside on their own backend server. This server has no wider access outside of the LAN. The various different embedded devices operate on the LAN, but may or may not require internet access depending on their make, and purpose. So for example, a non commercial range or temperature sensor, would be operating locally, but a commercial device like a Google Home, would require access to cloud services.

A travel router in this design has been employed to offer additional flexibility when IoT devices are not allowed to work on a wider area network. This, essentially is a router setup specifically for IoT networked devices, which accounts for common issues faced by, and reported about, IoT devices in general, such as in [17]. It also allows for use of common operating system functions like secure shell (SSH), or use of particular ports, and a visualization dashboard, which are sometimes blocked at the institution level. It further separates LAN traffic from the institutional network, which is often shared by many unknown devices, which would also be able to access the API endpoint if it were not in place. This keeps any possible tunnel compromises to the LAN, and helps mitigate positioning leaks, such as those highlighted in [36].

Additionally, the RaspberryPi (Pi) unit is acting as a host to particular sensors like a camera module, and also as a central point for other room devices to publish their sensor data. These devices can accomplish this by publishing data in an MQTT channel that can be subscribed to by application devices. The idea here is to distribute the activity on the network, so that the backend server is not handling every request or activity. This provides some flexibility in terms of data availability, as devices can subscribe to the Pi for room based sensor data, and access the the models at the same time. If, for example, a device was locally running its own agent on its hardware and did not need the models provided by the API, but did need the data from the room, it could do this without adding extra load to the backend server. Further, this approach gives the ability to run local webhooks for web service fulfillment, such as via Amazon Alexa, and gives a spot to run an NGrok tunnel without running it on a main backend server.

Figures 4 and 5 present the flows of information within the network, which depends on each device. Most devices on the local area network can make direct requests to the API endpoint on the backend server to access the models. Devices like a Google home that depend on an external service, would use the main network and gateway to access their services, however the webhook fulfillment would rely on the tunnel.

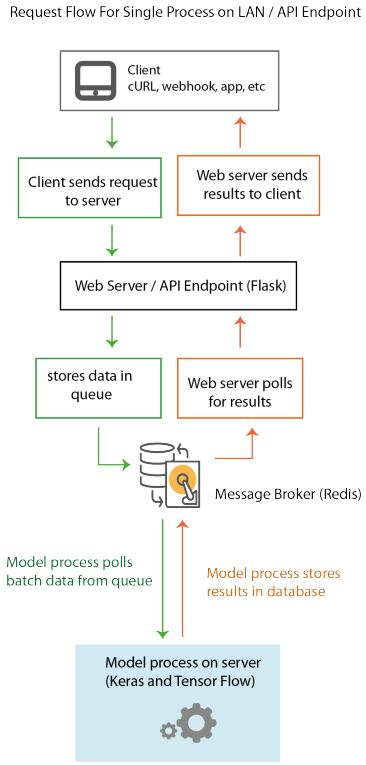

Figure 4: Request communication for a device on LAN setup.

5 DERIVING SECURITY AND PRIVACY CONSIDERATIONS FOR CORE IOT

The previous section highlighted the design of a prototype internet of things framework as a first step toward a capability which incorporates multiple perspectives, while allowing potential to bring multi-modal contextual information into the IoT system for sense-making, and in future toward potential visualization for users within the IoT system. The prototype addresses this core functionality, but has not been designed directly with privacy and security as a primary goal. This section aims to systematically understand the privacy and security landscape of the CORE system, leveraging a repeatable, methodological approach. Two design tools from literature have been considered, based on the work of [9] on deriving security considerations, on the one hand, and [18] for determining privacy design guidelines, on the other. To bridge these, the following steps are conducted, as in Figure 6.

The first step involves selecting and clarifying the CORE IoT components in order to situationally assess the security and privacy needs of each. The second step involves following elements of the OCTAVE-Allegro method where possible, to produce a set of security threats and risk profile scenarios. With an understanding of the risks faced by the CORE system in particular, the next step is to gain a similar understanding of privacy needs for this system, using Privacy-by-Design guidelines. These allow for the final step of deriving a combined set of security and privacy considerations for the CORE IoT prototype, as outlined within the following subsections.

5.1 OCTAVE-Allegro Methodology for CORE

The OCTAVE-Allegro risk assessment framework has been proposed in 2007 [9] as a fast and light-weight methodology based on the pre-existing OCTAVE methodology, applied by the military and large organizational teams, toward identifying and evaluating information security risks. This more recent refinement is targeted toward much smaller groups and individuals aiming to develop "broad assessment of an organization's operational risk environment...without the need for extensive risk assessment knowledge" [9]. This approach has recently begun to be investigated in the

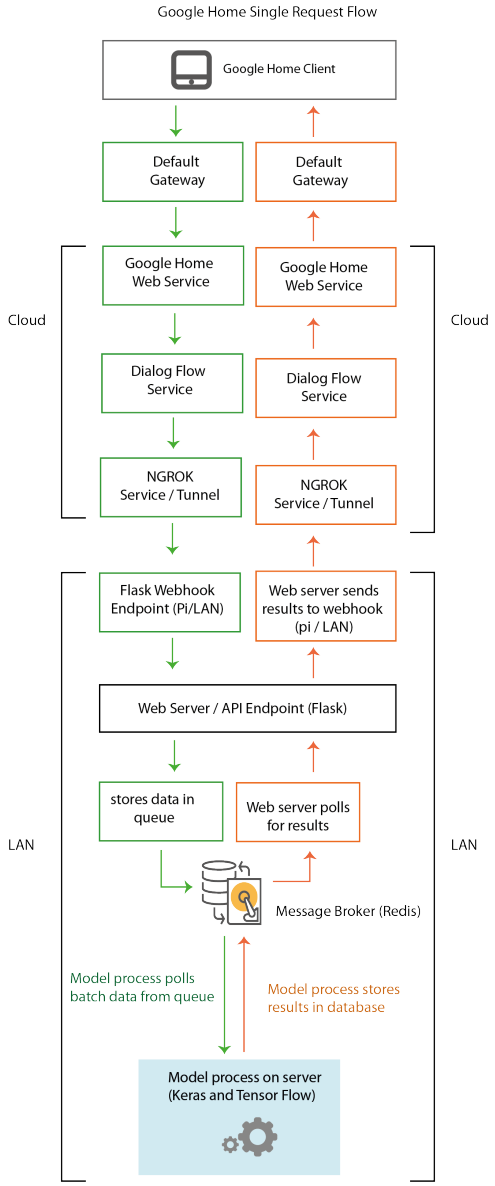

Figure 5: Request communication for cloud service devices.

IoT domain toward gaining an understanding of IoT security, as in [3]. The OCTAVE-Allegro methodology consists of eight steps toward four key tasks, namely establishing drivers, profile assets, identification of threats, and identification and mitigation of risks, [9].

In this work, a modified approach to the OCTAVE-Allegro has been conducted, with a main focus on providing a quick snapshot of the security profile of the CORE prototype. As such, it is OCTAVE-Allegro inspired, and focuses on four themes, as seen in Figure 7. The first step is to identify information assets for each component of the system. The next step involves identifying threats for each information asset identified. For each threat, potential mitigation

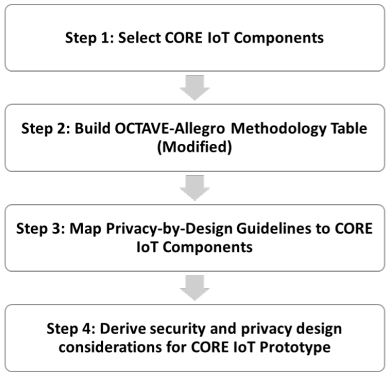

Figure 6: Methodology for deriving security and privacy considerations using OCTAVE-Allegro [9] and Privacy-by-Design [18] guidelines.

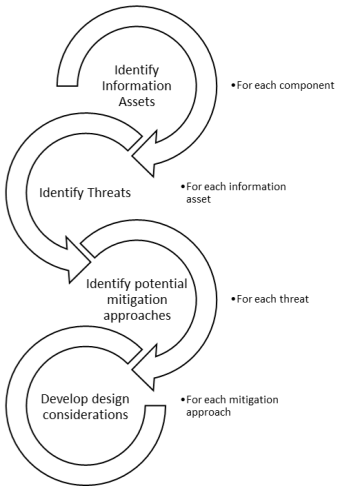

Figure 7: An OCTAVE-Allegro [9] inspired approach to understand security considerations.

approaches are selected, and lastly, for each mitigation approach a design consideration is developed.

While this shortens the OCTAVE-Allegro approach, it continues to provide useful tools to understand information assets and threats, and potential mitigation approaches, as in Figure 9.

5.1.1 Information Assets and Threats. In terms of the CORE system the information assets of each component has been identified, based on considering the relative value of the pieces of information within the component. For the *machine learning models* library this refers to the trained neural net models, the incoming data streams to these models, and the outgoing classification results from each model. For the *backend server* this refers to the model processing framework, the REST API endpoints for accessing models, the request queues or buffers (in Redis) for incoming service calls, and iv) the internal data stored on the server (in Redis). For the *embedded CORE service devices* this refers to audio camera or video feeds, sensory data, and the hardware related assets of each device, namely the IP address,

MAC address, or embedded processors. For *embedded commercial cloud service devices* this refers to details relating to account profiles, passwords, and the physical devices themselves which are allowed to connect to the varied services. For *cloud services* this refers to information related to accounts, service-specific data and API access points, as well as authorized device connections. For *network middleware*, this refers to information related to network traffic, routers and gateway connections. These information assets each require particular focus on the kinds of security events that may take place.

5.1.2 Threats and Potential Mitigation Approaches. With an understanding of the information assets within the system, a set of unique threats has been identified, and can be considered for the system in terms of security scenarios. This includes the following:

Altered models: Attackers may be able to enter the system and modify an ML model. This has a moderately high impact, but may be mitigated by performing validation checks on the model and its access. *Data theft:* Attackers may be able to steal copies of information assets. This has a moderately high risk, but can be mitigated by encryption techniques. *Data falsification:* Attackers may be able to inject falsified information into the system. This has a high risk, but may be mitigated by validating the source of information, via keys, or IP addresses, or even blockchain methods. *Data destruction:* Attackers may be able to delete information within the system. This has a high impact, but may be mitigated by making frequent backups of resources. *Identity theft:* Attackers may be able to access the system via authorized channels, or unsecured data channels, or by redirecting network traffic. This has a high impact but can be mitigated somewhat via decentralizing information channels and authorization endpoints. *Resource overruns:* Attackers may be able to overflow system information resource handling or buffers by spamming requests. This has a moderate impact, but may be mitigated by blocking offending traffic IP addresses, or using validation methods. *Hardware identity theft:* Attackers may be able to enter the network by using a compromised version of the system hardware. This is a high risk event, but may be mitigated by conducting validation checks, routine physical checks of hardware, and encryption techniques. *Malicious software:* Attackers may be able to use malicious software to gain access to network and information assets, or cause data losses. This is a high impact event, but may be mitigated via antivirus software and routine updates. *Hardware theft:* Attackers may gain access to physical hardware and any information assets on the device, or that can be accessed via the device. This is a high impact event, but may be mitigated by having better passwords, password security strategies, or encryption. *Access to cloud-based assets:* Attackers may gain access to cloud service infrastructures being used by the system. This has a significant impact, but may be beyond control, as the attacker is at the site of the cloud service center. Two-factor authentication may help to mitigate such an event. *Network infiltration:* Attackers may gain access to network traffic and may be able to read/decrypt information assets or packets. This is a moderate risk to system functionality, but may be mitigated by using strong encryption methods. Together, this set of scenarios highlights some of the central events that the CORE system designs must account for in terms of security.

	Privacy by Design Strategies from [18]
Minimise	"The amount of personal data that is processed should be restricted to the minimal amount pos-sible."
Hide	"Any personal data, and their interrelationships, should be hidden from plain view."
Separate	"Personal data should be processed in a distributed fashion, in separate compartments whenever pos- sible."
Aggregate	"Personal data should be processed at the highest level of aggregation and with the least possible detail in which it is (still) useful."
Inform	"Data subjects should be adequately informed when- ever personal data is processed."
Control	"Data subjects should be provided agency over the processing of their personal data."
Enforce	"A privacy policy compatible with legal require- ments should be in place and should be enforced."
Demonstrate	"Be able to demonstrate compliance with the pri- vacy policy and any applicable legal requirements."

Figure 8: Privacy by design strategies, as in [18].

5.2 Privacy-by-Design Strategies for CORE

Privacy within ubiquitous systems has been a long-held design consideration [22], but remains a current and important challenge for IoT system designers and engineers [28]. From a legal perspective, privacy has been the subject of much attention, as big data trends continue, and with more personal quantified data being gathered from users [32]. Recently, Privacy-by-Design strategies have been brought to the forefront [18, 32] toward providing inherent data protection, and the concept has had need of frameworks for elucidating the fuzzy privacy needs of systems. Frameworks like [28] are being tested to assess their impact on engineering more private systems.

In this work, the Privacy-by-Design strategies of [18] have been considered as a simplified starting point, as it is based closely on privacy and data protection laws and accounts for the needs of the software development life cycle. In particular, eight privacy design strategies were proposed, toward potential privacy design patterns. These are seen in Figure 8, and consist of strategies for i) Minimization of private information used by the system, ii) separation of private information within the system, iii) aggregation of private information when possible within the system, iv) hiding of private information from plain view within the system, v) informing users when private information is being used, vi) allowing users to have a measure of control over how or when this information is used by the system, vii) enforcement of a known privacy policy, and viii) demonstration of adherence to the agreed on privacy policy. With the exception of the latter two strategies related to privacy policies, this full set of privacy by design guidelines have been considered.

5.2.1 PbD and OCTAVE for CORE Components. Through cross-application of both the PbD strategies, and the OCTAVE-Allegro security assessment, it is possible to understand how the components of the system are impacted by specific security threats, on the one hand, but then how these same information assets and threats relate to privacy needs for each information asset in the system. This mapping is shown in Figure 9, and highlights the utility of this approach as a methodology toward merging privacy and security concepts. For each information asset, security considerations result from elicited mitigation strategies as in the OCTAVE-Allegro, while privacy considerations derive from cross-referencing threats to information assets per component with PbD strategies. This mapping together results in a listing of both privacy considerations

CORE IoT Component	Information Asset	Potential Threats	Unique Threat List	Minimise	Hide	Separate	Aggregate	Inform	Control
Machine Learning Models Library	Trained neural net models	Altering ML models	Altered models		X	X	X		
		Copying ML models	Data theft		X	X	X		
	Incoming data streams	Accessing incoming data (image) streams	Data theft	X	X	X		X	X
		Falsifying incoming data streams	Data falsification	X	X	X		X	X
		Copying incoming data streams	Data theft	X	X	X		X	X
	Outgoing classification result streams	Accessing outgoing result streams	Data theft		X	X	X		
		Falsifying outgoing result streams	Data falsification		X	X	X		
		Copying outgoing result streams	Data theft		X	X	X		
Backend Server	Model processing framework	Deleting server systems/infrastructure wipe	Data destruction		X				
	REST API endpoints	Modifying/re-directing API endpoints	Hardware identity theft		X				
	Request queues (Redis)	DDOS denial of service attacks	Resource overrun						
	Internal database (Redis)	Accessing internal server logs and databases	Data theft	X	X	X			
		Deleting internal server logs and databases	Data destruction		X	X			
Embedded CORE Service Devices (e.g., Pi, Photons, sensors)	Audio data	Stealing, falsifying, or copying data	Data theft, Data falsification	X	X	X	X	X	X
	Camera data	Stealing, falsifying, or copying data	Data theft, Data falsification	X	X	X	X	X	X
	Sensor data	Stealing, falsifying, or copying data	Data theft, Data falsification	X	X	X	X	X	X
	IP addresses	Spoofing or falsifying IP addresses	Hardware, identity theft, Data theft,		X				
	Physical device	Replacing physical unit with malicious unit	Hardware theft, Hardware identity theft		X			X	X
		Script-based hacking	Malicious software		X	X	X		
	MAC addresses	Spoofing machine addresses	Hardware identity theft		X				
	Embedded processor	Infiltrating hardware for Botnet uses	Malicious software		X				
Embedded Commercial Cloud Service Devices (e.g., Alexa, Wemos, Phone)	Account/Profile access	Stealing, falsifying, or copying profiles and data	Data theft, Data falsification		X				
	Passwords (for user)	Cracking passwords	Access to cloud-based assets		X				
	Device Hardware	Stealing data	Access to cloud-based assets		X			X	X
	Phone	Stealing physical device	Hardware theft		X			X	X
Cloud Service	Account	**same as embedded**	Access to cloud-based assets, Data theft, data falsification		X				
	Service data	Stealing, falsifying, or copying data	Data theft, data falsification		X				
	API endpoints	API hacking and port forwarding	Hardware identity theft, Data theft		X				
	Connection to devices (OAuth)	Account hacking	Access to cloud-based assets		X			X	X
Network Middleware	Traffic	Accessing network traffic (listening/packet-sniffing)	Network infiltration	X	X	X	X		
		Spoofing or falsifying identities (DNS, IP spoofs)	Hardware identity theft	X	X	X	X		
	Router	Hacking router	Network infiltration		X				
	Gateway Connections (Tunnelling/Ngrok)	Attacking connectivity resources, tunnel hijacking, TCP attacks, port forwarding	Network infiltration		X				
	Internal network gateway to ISP	Social engineering attacks at ISP	Identity theft						

Figure 9: Mapping Privacy-by-Design and OCTAVE-Allegro security needs for CORE IoT components.

and security considerations, and can be clearly identified as design requirements, as in Figure 10, for the CORE prototype.

Specifically, a set of 16 unique privacy considerations have been derived for the system, and 22 security considerations. These refer directly to Figure 9, and introduce aspects of design which, if possible and practical, the system should incorporate. This set of considerations also lends itself toward potential system evaluations, as it can be seen clearly how well the system meets each design consideration. As such, the current version of the CORE system has a resulting set of directions toward making privacy and security enhancements.

6 DISCUSSION

The development of an IoT system leverages different core components and tools, each with their own frameworks and API functions, communication channels, and protocols. It is important that systems designers have tools to consider the security needs of such systems, and the impact of systems on privacy needs of users, whether individuals or organizations. The methodology applied here shows an approach toward understanding the security risks that impact such a system, as well as the privacy risks. When combined this represents a useful toolset for system engineers. In many cases,

Num	Privacy Consideration (Where possible/practical, the system should…)	Addressed in CORE? (Y/N)
C1	Hide, separate, and aggregate private personal information related to trained neural net models to counteract altered model events and data theft events.	N
C2	Minimise, hide, and separate private personal information related to incoming data streams, and also inform individuals when this information is accessed, and allow users to control amount of information usage to counteract data theft events and data falsification events.	N
C3	Hide, separate, and aggregate private personal information related to outgoing classification result streams to counteract data theft and data falsification events.	N
C4	Hide private personal information related to model processing framework to counteract data destruction events.	N
C5	Hide private personal information related to REST API endpoints to counteract hardware identity theft events.	N
C6	Minimise, hide, and separate private personal information related to internal database to counteract data theft and data destruction events.	N
C7	Minimise, hide, separate, and aggregate private personal information related to audio data, and also inform individuals when this information is accessed, and allow users to control amount of information usage to counteract data theft or data falsification events.	N
C8	Minimise, hide, separate, and aggregate private personal information related to camera data, and also inform individuals when this information is accessed, and allow users to control amount of information usage to counteract data theft or data falsification events.	N
C9	Minimise, hide, separate, and aggregate private personal information related to sensor data, and also inform individuals when this information is accessed, and allow users to control amount of information usage to counteract data theft or data falsification events.	N
C10	Hide private personal information related to IP addresses to counteract hardware identity theft or data theft.	N
C11	Minimise, hide, separate, and aggregate private personal information related to physical devices to counteract hardware theft, hardware identity theft, or malicious software events and inform them of unauthorized accesses to information and allow them to control information usage.	N
C12	Hide private personal information related to MAC addresses to counteract hardware identity theft events.	N
C13	Hide private personal information related to Embedded processors to counteract malicious software events.	N
C14	Hide private personal information related to account profiles, logins, passwords, to counteract data theft, and data falsification events and unauthorized access to cloud-based assets.	N
C15	Hide private personal information related to service data, API endpoints, and device connections in order to counteract data theft, data falsification, and hardware identity theft events.	N
C16	Minimise, hide, separate, and aggregate private personal information related to network traffic, routers, and gateway connections (tunnels), to counteract network infiltration and hardware identity theft events.	N

Num	Security Consideration (Where possible/practical, the system should…)	Addressed in CORE? (Y/N)
S1	Perform validation check on model access.	N
S2	Encrypt incoming data streams.	N
S3	Perform validation check on incoming data streams.	N
S4	Perform validation check on outgoing classification streams.	N
S5	Encrypt outgoing classification streams	N
S6	Make regular backups of server.	Y
S7	Decentralize authorisation endpoints	N
S8	Block or minimize offending traffic IP's, or perform validation checks on access of server.	N
S9	Encrypt audio data	N
S10	Encrypt camera data	N
S11	Encrypt sensor data	N
S12	Perform validation check on IP addresses	N
S13	Perform physical inspection of devices and regular validation checks in software for physical devices, and encrypt sensitive information	Y
S14	Run antivirus software regularly on embedded devices	N
S15	Strengthen passwords/cycle passwords, and update operating system regularly.	Y
S16	Strengthen passwords/cycle passwords, use two-factor authentication	N
S17	Update or generate OAuth keys regularly	N
S18	Encrypt network traffic	N
S19	Rotate or update IP addresses regularly	N
S20	Apply multiple router solutions	Y
S21	Use multiple approaches, like HTTP Authorisation, and traffic validation checks, and apply secondary router, and rotate ports; potentially avoid use of tunnel completely, if possible	N
S22	Perform routine checks with ISP regularly	N

Figure 10: Privacy and security considerations discovered for the CORE IoT prototype, for future system improvement.

however, IoT developers may not consider such techniques, either by design or as an oversight, but the risks remain.

The proposed approach has leveraged two existing methods for understanding security and privacy, and has derived a set of design considerations for the specific CORE prototype. A comparison of this with existing techniques is a clear avenue for future research. Also, it remains to be seen whether these design considerations result in a more secure and private system in practice, and whether the development costs of implementation of these concerns is feasible and beneficial. These are two important aspects for further research, and future work will aim to systematically assess the impact of introducing the security and privacy considerations derived in this work. It is hoped that this will allow for a more secure and private system, although this remains for further exploration. However, as researchers continue to consider the approaches combined in this work, like [28] for Privacy-by-Design in IoT, and [3] for IoT security with OCTAVE-Allegro, it is expected that this merger will aid developers.

6.1 Maker Perspectives and Security

"You have a 9/10 chance that somebody already made it and that it's posted somewhere on the site. You would be dumb not to use it" [10].

The approach to this prototype is brought about from a more maker perspective where, for this sake of this paper, a Maker can be defined as someone who participates in Maker subculture and general Do-it-Yourself (DiY) movements, as in [34]. The DiY culture refers to a societal movement of doing and making things oneself, originating from dissatisfaction with current society [10]. Making is generally seen as a democratization of techniques and technologies [34], and Makers are interdisciplinary people, who participate in a lot of different communities from crafters, and hobbyists, to hackers and professionals [34]. A maker approach can be thought of as relying on experimentation and having a prototyping-first mindset. While traditionally a software development approach to implementing connected devices can be thought of as a plan first approach, a maker approach tinkers first, re-working an unfinished project, using existing materials to inform creation, or building on projects or creation processes that have already been done in the past [10]. One issue is that maker related tools, and discussion, do not really focus on security of those tools, as security is often at odds with experimentation and sandboxing.

This means that maker approaches and entry points as outlined by Dries De Roeck et al [10], are sometimes hampered, or put into

a position of only existing purely externally in the cloud, or completely locally (which is not always possible). IoT projects can now involve many different kinds of devices, as shown above in even a basic prototype, that can span from DIY based board, and sensors, to commercial products which can have requirements that are not always in the best interest of IT security. In the case of the CORE Network, the current prototype was developed by leveraging existing tutorials, open source software libraries, and following the examples of implementation provided by existing community members. The entry point for the project, was a combination of considering what hardware elements already existed at arm's length, and how they could be utilized together. Hence, security and privacy enabling strategies like the one here proposed have utility in allowing system engineers, especially makers, to clearly assess design decisions and the systems that result.

7 SUMMARY AND FUTURE WORK

The internet of things represents a systems engineering paradigm with inherent privacy and security challenges due to its growing ubiquitous and decentralized components, service-oriented architectures, and embedded systems frameworks. It is increasingly imperative for IoT systems designers and engineers to have tools that help to clarify the privacy and security needs of these systems throughout their lifecycles. This work has aimed to explore and address this need by presenting a new IoT system design, aimed at an indoor smart environment that leverages information from multimedia sources, such as sensing and computer vision, and that aims at maintaining a human-centered approach. The security and privacy of this system has been derived using a merger of a risk-assessment methodology for security considerations with a set of privacy-by-design strategies. This has resulted in a structured set of both privacy and security considerations for the proposed system. The next stages of this work will aim to assess whether the implementation of these privacy and security considerations, where practical and possible, can enhance the existing system in practice. It is hoped that future researchers will leverage similar techniques for a wide range of IoT systems.

REFERENCES

[1] Emile Aarts. 2004. Ambient intelligence: a multimedia perspective. *MultiMedia, IEEE* 11, 1 (2004), 12–19.

[2] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. 1999. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*. Springer, 304–307.

[3] Bako Ali and Ali Ismail Awad. 2018. Cyber and Physical Security Vulnerability Assessment for IoT-Based Smart Homes. *Sensors* 18, 3 (2018), 817.

[4] Amazon.com. 2018. Alexa Skills Kit - Build for Voice with Amazon. Retrieved July 17, 2018 from https://developer.amazon.com/alexa-skills-kit

[5] Amazon.com. 2018. Echo and Alexa - Amazon Devices. Retrieved July 17, 2018 from https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011

[6] Apache.org. 2018. The Apache HTTP Server Project. Retrieved July 17, 2018 from https://httpd.apache.org

[7] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The internet of things: A survey. *Computer networks* 54, 15 (2010), 2787–2805.

[8] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2017. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks* 56 (2017), 122–140.

[9] Richard A Caralli, James F Stevens, Lisa R Young, and William R Wilson. 2007. *Introducing octave allegro: Improving the information security risk assessment process*. Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[10] Dries De Roeck, Karin Slegers, Johan Criel, Marc Godon, Laurence Claeys, Katrina Kilpi, and An Jacobs. 2012. I would DiYSE for it!: a manifesto for do-it-yourself internet-of-things creation. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*. ACM, 170–179.

[11] Flask.pocoo.org. 2018. Flask-a Python Microframework. Retrieved July 17, 2018 from https://flask.pocoo.org/

[12] Github.com. 2018. Pose Detection in the Browser: PoseNet Model. Retrieved July 17, 2018 from https://github.com/tensorflow/tfjs-models/tree/master/posenet

[13] Google.com. 2018. Google Assistant - Your own personal Google. Retrieved July 17, 2018 from https://assistant.google.com/intl/en_ca/

[14] Google.com. 2018. Google Cloud Platform. Retrieved July 17, 2018 from https://cloud.google.com

[15] Google.com. 2018. Google Home - Smart Speaker and Home Assistant. Retrieved July 17, 2018 from https://store.google.com/product/google_home/

[16] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems* 29, 7 (2013), 1645–1660.

[17] Dominique Guinard and Vlad Trifa. 2016. Building the Web of Things: With examples in Node. js and Raspberry Pi. (2016).

[18] Jaap-Henk Hoepman. 2014. Privacy design strategies. In *IFIP International Information Security Conference*. Springer, 446–459.

[19] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. 2009. Context-aware systems: A literature review and classification. *Expert Systems with Applications* 36, 4 (2009), 8509–8522.

[20] Keras.io. 2018. The Keras Blog. Building a simple Keras + deep learning REST API. Retrieved July 17, 2018 from https://blog.keras.io/building-a-simple-keras-deep-learning-rest-api.html

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[22] Marc Langheinrich. 2001. Privacy by designâĂŤprinciples of privacy-aware ubiquitous systems. In *International conference on Ubiquitous Computing*. Springer, 273–291.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[24] Alexis Morris. 2009. *Socio-technical systems in ICT: A comprehensive survey*. Technical Report. University of Trento.

[25] MQTT.org. 2018. MQTT. Retrieved July 17, 2018 from https://mqtt.org/

[26] Ngrok.com. 2018. Ngrok-Secure Introspectable Tunnels to Localhost. Retrieved July 17, 2018 from https://ngrok.com/

[27] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. 2017. Towards accurate multi-person pose estimation in the wild. In *CVPR*, Vol. 3. 6.

[28] Charith Perera, Mahmoud Barhamgi, Arosha K Bandara, Muhammad Ajmal, Blaine Price, and Bashar Nuseibeh. 2017. Designing Privacy-aware Internet of Things Applications. *arXiv preprint arXiv:1703.03892* (2017).

[29] RaspberryPi.org. 2018. RaspberryPi - Teach, Learn, and Make with Raspberry Pi. Retrieved July 17, 2018 from https://www.raspberrypi.org

[30] Redis.io. 2018. Redis. Retrieved July 17, 2018 from https://redis.io/

[31] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. *arXiv preprint* (2017).

[32] Peter Schaar. 2010. Privacy by design. *Identity in the Information Society* 3, 2 (2010), 267–274.

[33] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[34] Joshua G Tanenbaum, Amanda M Williams, Audrey Desjardins, and Karen Tanenbaum. 2013. Democratizing technology: pleasure, utility and expressiveness in DIY and maker practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2603–2612.

[35] TensorFlow.org. 2018. TensorFlow - An open source machine learning framework for everyone. Retrieved July 17, 2018 from https://www.tensorflow.org/

[36] Tripwire.com. 2018. Google's Newest Feature: Find My Home. Retrieved July 17, 2018 from https://www.tripwire.com/state-of-security/vert/googles-newest-feature-find-my-home/

[37] Wemo.com. 2018. Wemo Smart Bridge. Retrieved July 17, 2018 from http://www.wemo.com

[38] Alexander Wong, Mohammad Javad Shafiee, Francis Li, and Brendan Chwyl. 2018. Tiny SSD: A Tiny Single-shot Detection Deep Convolutional Neural Network for Real-time Embedded Object Detection. *arXiv preprint arXiv:1802.06488* (2018).

[39] Mark Zack. 2014. The fundamental components of the Internet of Things. Retrieved July 17, 2018 from https://www.electronicsworld.co.uk/news/advertorials/5022-the-fundamental-components-of-the-internet-of-things