Big ideas paper: Policy-driven middleware for a legally-compliant Internet of Things

Jatinder Singh
Computer Laboratory
University of Cambridge, UK
js573@cam.ac.uk

Thomas Pasquier
CRCS
Harvard University, USA
tfjmp@seas.harvard.edu

Jean Bacon
Computer Laboratory
University of Cambridge, UK
jmb25@cam.ac.uk

Julia Powles
Faculty of Law
University of Cambridge, UK
jep50@cam.ac.uk

Raluca Diaconu
Computer Laboratory
University of Cambridge, UK
rd530@cam.ac.uk

David Eyers
Dept. of Computer Science
University of Otago, NZ
dme@cs.otago.ac.nz

ABSTRACT

Internet of Things (IoT) applications, systems and services are subject to law. We argue that for the IoT to develop lawfully, there must be technical mechanisms that allow the enforcement of specified policy, such that systems align with legal realities. The audit of policy enforcement must assist the apportionment of liability, demonstrate compliance with regulation, and indicate whether policy correctly captures legal responsibilities. As both systems and obligations evolve dynamically, this cycle must be continuously maintained.

This poses a huge challenge given the global scale of the IoT vision. The IoT entails dynamically creating new services through **managed and flexible data exchange**. Data management is complex in this dynamic environment, given the need to both control and share information, often across federated domains of administration.

We see middleware playing a key role in managing the IoT. Our vision is for a middleware-enforced, unified policy model that applies end-to-end, throughout the IoT. This is because policy cannot be bound to things, applications, or administrative domains, since functionality is the result of composition, with dynamically formed chains of data flows.

We have investigated the use of Information Flow Control (IFC) to manage and audit data flows in cloud computing; a domain where trust can be well-founded, regulations are more mature and associated responsibilities clearer. We feel that IFC has great potential in the broader IoT context. However, the sheer scale and the dynamic, federated nature of the IoT pose a number of significant research challenges.

Keywords

Law, regulation, policy specification and enforcement, audit

1 Introduction and context

The *Internet of Things* (IoT) is a subject of wide publicity and speculation. The ISO [4] describes the IoT as *"An infrastructure of interconnected objects, people, systems and information resources together with intelligent services to allow them to process information of the physical and the virtual world and react."* This broad vision embodies the concept of ubiquitous/pervasive computing: the seamless integration of sensor and actuation technologies into a wide-scale (global) systems environment, capable of reacting appropriately to changes in context to provide desired functionality.

The notion of the IoT is becoming mainstream: 'bringing the world online' impacts individuals, groups, organisations, governments, and society as a whole. The connected systems infrastructure is envisaged as supporting a vast number of different applications, in domains such as smart cities, healthcare, traffic monitoring, energy efficiency, and personal lifestyle management. All of these applications are intended to be customisable to specific needs. They also raise complex questions of responsibility regarding data, particularly given that sensors, IoT devices and their enabling systems bear constant witness to our everyday lives.

At the same time, the IoT is a subject of increasing political and legal interest [38], not least because of the scale of personal and/or otherwise sensitive data that it entails, raising a range of private law issues, particularly under data protection, privacy, contract and tort law. This is what we envisage by "legally-compliant IoT". For the avoidance of doubt, our motivation is not to instrument the IoT for criminal investigations or national security purposes. Surreptitious actions, such as those by malicious parties and government agencies, are beyond the scope of this paper.

Fig. 1 shows the big picture. Law and regulation, reflecting responsibilities and obligations, together with personal preferences, must be embodied in *policy*, which technical mechanisms must enforce system-wide. Such policy must be continually aligned with evolving law and regulation, and the audit of its *enforcement*, particularly regarding data flow and processing, is necessary to demonstrate compliance. Following this model, which is far from current practice, the **"big idea"** we put forward is that a legally-compliant IoT is a real possibility, enabled by decentralised policy enforcement on a middleware foundation. Middleware is an ideal basis for such functionality, as it operates across system components in an application-independent manner.

Most current middleware focuses on enabling functionality. Our idea is that middleware must go further, to enable distributed compliance. This area is relatively unexplored by the middleware community, but is crucial to moving forward, particularly given the emerging regulatory landscape.

1.1 Dynamic processing chains in the IoT

Realising functionality in the IoT potentially involves long processing chains, comprising many system components (see Fig. 2), often representing a *system-of-systems*. IoT architectures include physical devices (sensors, actuators); gateways

Pre-print for ACM/IFIP/USENIX Middleware 2016 (http://dx.doi.org/10.1145/2988336.2988349)

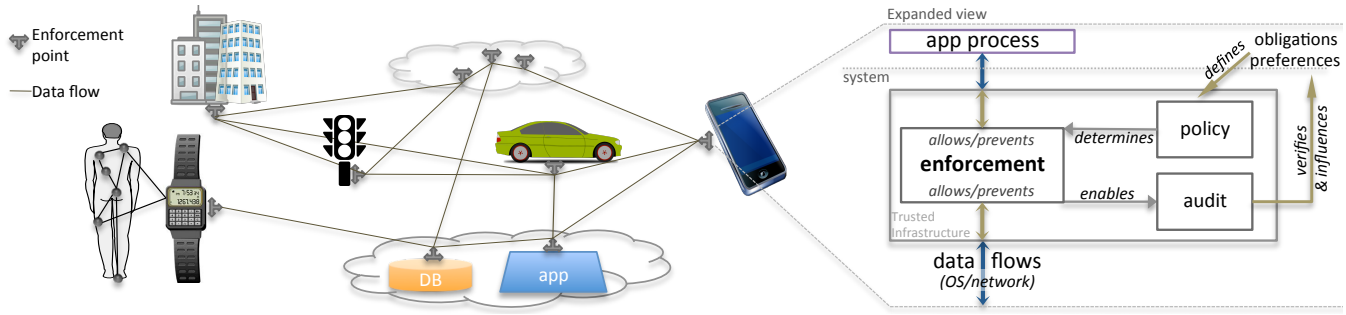

Figure 1: A high-level overview of decentralised policy enforcement in the IoT

(hubs, phones, domain administration agents); and private/hybrid/public cloud services, for storage, processing and analytics. All might be managed by different parties, with different interests and obligations. The chain of components delivering functionality might vary, even for instances of the same application; e.g. a mobile application may interact directly with 'things' in the local environment, which vary according to location. Further, the same components can be used for different purposes, in different circumstances.

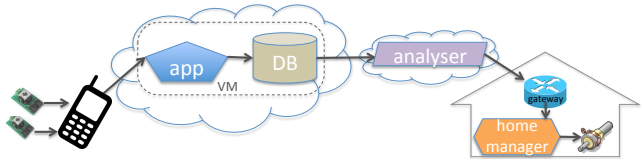

Figure 2: An IoT functional component chain

In the IoT, management policy relates to personal or organisational preferences, regulatory and contractual requirements, and context. Given the above, where system components can be (re)used and *interact* (exchange data) in unforeseen and customised ways, these policies must be ascertained, respected and enforced throughout the highly dynamic environment. This requires supporting infrastructure to provide the means for policy to set constraints and reconfigure system components, to ensure that interactions properly accord with higher-level goals and obligations.

1.2 Data-centric access control, end-to-end

Traditional access controls (authentication and authorisation) tend to operate only within the scope of a particular application/system, with enforcement only at specific points. While these controls are needed in the IoT, it is also necessary for system infrastructure to enable the control and management of data, end-to-end, under a consistent, continuously enforced policy regime, as data moves along any processing chains. Information Flow Control (IFC) augments traditional access control by providing continuous, system-wide, end-to-end flow control based on labelled properties of the data. We propose that IFC should be vigorously investigated for use in the IoT to complement existing security regimes, as explained further in §6 and §7.

Data analytics are an important part of the IoT, for managing context (*"detect/respond"* event-based architectures), and enabling new services and efficiencies. This is another reason why *data-centric controls* are needed (in addition to application/system-centric controls) that also facilitate data provenance tracking. Once IFC is deployed, audit can easily be supported since a record can potentially be made of every attempted data transfer or access [68].

1.3 Summary and big idea

The IoT's growth will be restricted unless a means is developed to allow parties to meet their legal obligations. We argue that policies for data control must be consistently applied throughout the IoT, reflecting individual preferences, rights, regulation and law. This requires data sharing to be controlled both locally and within application/domain boundaries, as well as when data flows through IoT system chains (end-to-end). Since policy is often context-aware, and can change/evolve, systems will require reconfiguration to ensure that obligations continue to be met, under changing circumstances. Audit, traceability and transparency tools that can operate system-wide are important for accountability and demonstrating compliance. As regulation increases for the IoT it becomes clear that a feedback loop via technically-enforced policy is needed, as shown in Fig. 1.

Our big idea is for a legally-compliant IoT, where the supporting infrastructure enables the enforcement of policy to allow parties to demonstrably meet their responsibilities.

Middleware can support secure, managed (i.e. driven by policy), data sharing [82]. There is a clear role for middleware that enables dynamic, external reconfiguration, allowing management policy to be applied within the federated, decentralised and long-lived systems environment of the IoT. As a starting point, we feel that IFC, which is a data-centric security technology for continuous data flow control, shows promise in enabling end-to-end enforcement. IFC is not a new concept, but has not been applied in wide-scale distributed systems.

Clearly, any big idea must be grounded in experience with technologies that show promise. In previous work [68,69] we have demonstrated the potential of IFC in a cloud context; however, extending the approach to the IoT is challenging, given that cloud often represents a single administrative domain where trust can be well-founded, regulations are relatively mature and associated responsibilities are clear.

In this paper, we present early ideas and challenges concerning how IFC concepts, coupled with a reconfigurable, distributed systems middleware could apply in heterogeneous, untrusted, dynamic, and wide-scale systems environments, as a means towards enabling a legally-compliant IoT.

2 IoT architecture

The term IoT is used broadly, in a variety of contexts, from communication mediums, sensor networks and pervasive computing, to HCI and more. We have highlighted the need for dynamic, yet controlled, secure data sharing to realise the

broad IoT vision. This means that data flows must be managed, even in the presence of an ever-greater number of system components (representing communication endpoints)—so-called *hyperconnectivity*—and the increasing instrumentation of the physical world through ongoing improvements in and deployments of sensor/actuator technology.

Our discussion is therefore in the context of *interactions* (communication) between 'things', as shown in Fig. 1. In line with the ITU [2] and ISO [4] definitions, we use *thing* to refer to an entity, physical or virtual,[1] capable of interaction in its own right; thereby encompassing sensors, devices, applications/services (standalone or cloud-hosted), gateways, etc. Much research has been carried out on communication protocols for the IoT (e.g. to support low-powered devices in an IoT context) and software is becoming available, such as 6LoWPAN [57], COaP [80], etc. Policy at this lower-level concerns communication and/or resource constraints; our focus here is on higher-level data sharing.

2.1 Subsystems and gateways

Though many entities capable of communication make up the IoT, not all operate as "first-class citizens", because:
(1) They may be part of a subsystem that is bespoke or closed, rather than open or interoperable, e.g. as part of a legacy system or because of the manufacturer's policy;
(2) They may be extremely resource limited; or
(3) They may be part of a self-contained administrative domain or network, limited in scope, e.g. a workplace, a proprietary sensor network or industrial control system, or simply running behind a firewall [85]. This can be dynamic, in that networks may be *ad hoc* (e.g. mesh-networks), and 'things' can be mobile, e.g. someone's phone being directly accessible when in public (through their mobile provider), but behind a gateway when connected to an internal network at work.

We consider such entities to be part of *subsystems* that participate in the IoT via *gateways*, see Fig. 2. Gateways manage system boundaries, operating as 'hubs' that manage interactions on behalf of the subsystems they front, and/or providing capability (and policy) management services appropriate for the local environment. We therefore consider such gateways as 'things', as they represent a point in which policy can be enforced. Conversely, a single device could be considered as several 'things', e.g. a phone can host several applications, each capable of direct, external communication, and therefore subject to separate policy regimes.

2.2 Cloud services within the IoT

Cloud services are increasingly playing a role in supporting the IoT. A recent survey showed 34 out of 39 IoT architectures investigated included cloud and/or centralised services in some form [55]. Cloud services can be used:
(1) For processing and storage, especially when 'things' are low-powered and/or mobile; for archiving, and for aggregating sensor and other data and performing analytics;
(2) To allow IoT data to be integrated with more traditional services such as electronic health records;
(3) To operate as a coordinator/mediator, managing and controlling 'things' as appropriate to context.

There are ongoing issues regarding the security of cloud computing [90]. In [85] we discuss the security issues specific to cloud-supported IoT. Our current research is exploring the potential of IFC to provide data protection with controlled sharing in cloud computing, i.e. to manage intra- and inter-cloud interactions. We outline this work in §8, as a basis for exploring IFC for securing IoT data flows.

Any work aiming to incorporate cloud services into IoT must take account of emerging cloud developments, such as 'cloudlets' [78] and 'droplets' [26] that enable smaller, mobile, and personal/application-specific clouds. These aim at supporting emerging IoT environments, and can simply be considered 'things' for the purposes of our discussion.

3 Need for managed data sharing

Much IoT research has focussed on specific applications, and/or the technical aspects in bringing 'things' online, e.g. protocols, radio and power management, etc. There has been comparatively little attention on user-level security concerns (or other aspects of control) *across ranges of 'things'*, particularly relating to managing data sharing in this dynamic world, where 'things' can be (re)used to realise new, possibly unforeseen functionality. Six concerns are of particular relevance when considering IoT data sharing.

Concern 1: IoT involves personal and sensitive data. In the IoT, sensors will generate volumes of data relating to many aspects of life. Indeed, highly personal data will drive many envisaged IoT services. In contrast with traditional IT, the data sources may be more diverse, *ad hoc*, mobile and unmanaged. There is a great deal of law and regulation, national and regional, focussing on personal data [54], and the need for a sound legal basis (often, explicit consent) for the collection, maintenance and use of information that is identifiable to an individual. Commercially sensitive data may also arise in the IoT; again, regulatory and contractual constraints may apply. Determining responsibility and demonstrating compliance remains an ongoing issue [86].

Concern 2: IoT involves actuation. Actuation entails physical, real-world impact. Error, malice or mismanagement of actuation data flows (commands) can be catastrophic, and naturally entail legal consequences [35]. Flows may be to vehicles and smart homes, and also software updates for devices.

Concern 3: Policy persistence through chains. Because component interactions can be dynamically orchestrated, and IoT-component chains can be long (Fig. 2), it can be uncertain where data has flowed and how it is used. Once data has left one's control—i.e. it is 'out of their hands'—it is hard to trace. Generally, policies should persist throughout the data's entire lifecycle. This also helps with the amalgamation of data with different policies. To demonstrate that policies have been respected it is necessary to record and audit the flow of data.

Concern 4: Data quality and provenance. Quality of data relates to its source, e.g. user input, data from an approved sensor, any intermediate processing operations, and whether it is standards-compliant. Provenance concerns making data lifecycles visible. Again, this makes it necessary to record the flow of data.

Concern 5: Data aggregation, inference and analytics. Data in the IoT is diverse, coming from different users and sources. Data may be benign in isolation, but sensitive when combined; e.g. an anonymisation algorithm may yield data that, when combined with other data, leads to individual reidentification [62]. That is, subtle, unforeseen inferences can arise. Data analytics (often on 'big data') is

[1] Many definitions include 'virtual' aspects, as the delivery of IoT services involves a number of software components.

commonplace, e.g. to profile people's spending habits. IoT increases the scope for data analytics, but with less visibility over the outcomes [62]. Such issues motivate the need for data flows to be managed and visible, system-wide.

Concern 6: Customisation and Context. 'Things' will be used in different ways, for different purposes, e.g. a television that monitors viewing habits, designed for targeted advertising, may also allow a clinician to monitor someone's mental state. That is, the way 'things' are coordinated, i.e. how and when data moves between 'things', can be customised to achieve particular functional goals. Such concerns are context-dependent, e.g. the data from someone's medical monitoring devices may flow only to their local personal applications and storage services. In an emergency, "break-glass" policy overrides normal security constraints, alerting emergency services and (say) a family member, and replugging the sensor-data streams to make them available to the emergency response team [81]. More generally, perhaps a nurse should be able to access patients' data only when detected in the context of their homes. In both examples, security levels are adjusted—*reconfigured*—to accord with changes in context. In practice, many adaptations will be based on location and the local environment.

3.1 Policy for managing data

The IoT requires these concerns to be managed, in accordance with preferences and requirements. *Policy* encapsulates a set of concerns, defining the actions to take in particular circumstances to effect some outcome. Policy is often represented in rules and processes governing system behaviour, defining the actions to take when situations arise.

Much systems research on policy concerns lower-level aspects [88], such as resource allocation and/or quality-of-service in network management. In contrast, in an IoT architecture, policy *must also* capture higher-level (user) concerns, because these drive the interactions involving data exchange, and **must reflect the laws and regulations that apply.** Moreover, policies must be continuously aligned with changing regulations and user preferences. Therefore, to realise the IoT vision, policies must be dynamic in nature, operating to regulate data exchanges. This may involve enforcing policies at various levels of the technology stack (see §8.2.2). It is also important that auditing occurs so that compliance with policy can be demonstrated and accountability established. Related work on policy is summarised in §10, but there remain a number of open challenges regarding the scale and scope of the IoT, as this paper explores.

4 Common security approaches

We now consider the extent to which the most common "off-the-shelf" security mechanisms can meet the data sharing concerns just described. §10 considers less established work.

Access control (AC) comprises *authentication* (identity) and *authorisation* (the right to perform an action). For the IoT, authentication schemes should apply system-wide (see §9), meaning certificate-based (PKI) models could be used. Authorisation policy might target a particular entity, a role, and/or some aspect of context, e.g. parametrised roles can capture details of an entity, its functionality and context [10].

AC clearly plays an important part in governing systems. However, there are two related considerations for AC in IoT:

1. ACs are applied at specific *Policy Enforcement Points* (PEPs), authenticating and authorising a particular *action* (e.g. data exchange). While this protects the specific (peer-based) interaction, there is generally no subsequent control over data flows beyond the point of enforcement; and

2. AC tends only to be relevant within a particular application and system context, e.g. the AC of a database is different from that of the file system, and similarly they will vary depending on the entity, e.g. the AC of a cloud service provider targets different principals from those of the tenant's application it hosts. There are no continuous controls beyond the PEPs, nor guarantees that any data policy regime applies consistently throughout the infrastructure. For example, database tables may be shared between several applications. Although the applications enforce AC with their users, they may not have the same AC policies when operating on common data.

PKI. *Public Key Infrastructure* (PKI) can be useful in enabling a wide-scale security regime. One can envisage a PKI where 'things' have private keys and public key certificates, signed by a *certificate authority* linking them to their owners, who are also associated with certificates. Decentralised trust models (a *web-of-trust*) are also possible, assisting more *ad hoc* management by removing the need for a central authority. X.509 certificates [30] can be used for *authentication/identification* and *authorisation* [22] and are widely available and well understood. PKI provides a building block to be leveraged for a wider IoT infrastructure. Already, lightweight PKI schemes have been proposed (e.g. [41]) that could help support low-powered IoT devices.

Encryption. There is a clear role for encrypted communication channels e.g. via Transport Layer Security (TLS), or more lightweight versions for resource-challenged 'things' (§10). TLS relies on PKI and certificate authorities.

Application-level encryption (at data-item level) can secure data beyond the application's scope, but this raises a number of issues. First, key management (allocation, revocation and distribution) is non-trivial, particularly in the IoT where data restrictions change dynamically. Second, much of the IoT vision is of component chains that process, aggregate and analyse data. Encryption at the data-item level precludes certain processing services (e.g. in-cloud), unless keys are distributed. There is ongoing work on *homomorphic encryption* that will allow computation on encrypted data, but is not yet practicable for wide-scale deployment [40]. Finally, there is no logging/feedback on when data is decrypted. This makes it difficult to discover when, where and how data leaks occur, hindering audit and accountability.

Proxy re-encryption involves a semi-trusted proxy that transforms encrypted data produced by one party into a form decryptable by another, where the proxy cannot access the plaintext [7]. This allows third parties to manage the data of others, without having access to the content. This can shift key management overheads away from lightweight 'things', and potentially enables more secure orchestrations.

Differential privacy regulates the queries on a dataset and modifies result sets to balance the provision of useful, statistical-based results with the probability of identifying individual records [29]. This is useful for data analytics.

Hardware Based Trust. To establish trust in a system, the hardware cryptographic capability provided by *Trusted Platform Modules* (TPM) [11] is already used. Relevant here is how TPM can guarantee the integrity of a platform and its configuration, and also certify identity [70].

Intel SGX [3] provides an alternative to TPM-style trusted hardware, by providing means to run a trusted computing base in user space, above an untrusted OS kernel. Although SGX can facilitate confidentiality and integrity of data on a local machine, during external interaction the availability of the service cannot be protected from the underlying OS. ARM TrustZone [77] provides yet another design point, between the previous two, providing hardware-isolated, secure and insecure "worlds" in which the CPU operates. Integrity guarantees are particularly relevant in ensuring the security behaviour of a system, e.g. "can I trust this remote host to handle my data?" In an IoT context, such guarantees are applicable to 'things', e.g. helping to prevent impersonation and providing trustworthy information as to their specification and operational state, such as geographical location [44].

Also relevant is *remote attestation* [47], which provides the means to verify the integrity of a remote machine before interacting. Such technology is increasingly common, e.g. in virtual machines/containers on a cloud platform [17], embedded systems [5], and mobile phones [61].

Summary. These mechanisms provide an essential foundation on which to build the future IoT, with considerable IoT research (see §10) dedicated to extending them. But existing security techniques, by themselves, do not fully address the concerns of §3, which arise from one or more of: (1) the IoT's vast scale; (2) the variable, dynamic nature of the environment; and (3) the need for continuous enforcement, system-wide, as data flows span a range of services and administrative domains.

5 Middleware's role in the emerging IoT

We have discussed the requirements for controlled, dynamic data sharing in IoT and have considered security mechanisms of relevance. In this section we expand on the key role of middleware in IoT, with policy enforcement as the focus. Event-driven systems embody policy-driven behaviour [88]; for example, *Event-Condition-Action (ECA)* rules can specify the circumstances under which systems need to be reconfigured [87]. As mentioned, this was traditionally applied to network management. *Complex Event Processing (CEP)* engines have been developed for specific application areas, such as financial trading. Machine learning is gaining prominence, and can be used for learning and recognising significant patterns of events that can drive actions. Regardless of how policy is described and actions decided, our concern is the underlying mechanisms enabling policy to maintain appropriate system behaviour.

5.1 The need for policy-enforcing middleware

We have argued that policy is central to realising the wider IoT vision. Encoding policy in application logic is inherently limiting, as the wider vision entails components being (re)used, orchestrated and customised in various ways, including entirely novel uses. Policy that tends towards building silos hinders this wider vision.

Middleware (MW) is a layer of abstraction that mediates component interactions. As MW operates across components, it is the natural location for integrating data management mechanisms. This enables the means for application-related policy to operate system-wide. This is important in an IoT context because:

- Operating across 'things', MW can ensure the enforcement of a common, compatible management regime *throughout the IoT provisioning chain*; and
- In providing standard interfaces, MW allows 'things' to be developed without requiring policy to be defined *a priori*, helping to avoid unnecessary operational silos.

For developers and designers, middleware-based policy enforcement means they are not burdened with maintaining their own internal policy representation, thus removing the impossible requirement to account for all possible uses and operating environments for their component, and all its possible interactions. This is important for IoT, where 'things' are orchestrated to realise functionality. Also, policy management is made more tractable, since having fewer points of enforcement reduces the propensity for errors, see §5.2.

In short, middleware should provide the mechanisms for data security/management regimes to apply across systems, throughout the IoT system supply chain. This is crucial for realising the wider vision of the IoT.

5.2 Requirements of policy-driven MW

In considering policy enforcement, it is necessary to identify where control should be exercised. PEPs were introduced in §4 for access control. PEPs exist throughout the IoT (Fig. 1), with enforcement potentially occurring at each step of a component chain (Fig. 2). This adds complexity because:

- There is a need to enforce common policy at each control point along the entire path the data takes;
- As circumstances change, the policy at any of these control points may need to be updated, and the chains of flow may themselves need to change; and
- The decisions taken at each of these points must be made visible for audit and provenance-checking.

To elaborate, these requirements involve:

Continuous, end-to-end policy enforcement. Secure, reliable data sharing must be enforced end-to-end, throughout the entire service provisioning chain, including any intermediate sub-chains.

Dynamic, context-aware reconfiguration. The purpose of reconfiguration is to adapt the 'always-on' nature of the IoT to ensure compliant behaviour in changing circumstances. Here, reconfiguration may take the form of:

- Setting the security/management regime to disallow/allow particular data exchanges, i.e. setting up an environment, privileges, IFC security context (§6); or
- Proactively taking direct security operations, such as initiating/ceasing connections, e.g. forcing data through a sanitiser (§7), disconnecting an employee after their shift, or preventing a rogue 'thing' from causing more damage, e.g. by isolating or commands to shut it down.

In some circumstances, these may involve local reconfigurations, in others, a change in security preferences must be propagated through a range of different 'things'.

Data provenance and audit. Providing transparency and traceability by being able to track and audit the flow of data throughout the system, including the policies applied, reconfigurations initiated and interactions undertaken has several purposes: (1) ensure data quality; (2) help identify policy errors and encourage refinement; and (3) demonstrate compliance and aid accountability.

6 Information Flow Control (IFC)

So far we have described the need for system-wide data management for the IoT, and the need for policy-driven middleware. We now introduce IFC as a model for both tracking and constraining information flows. We show IFC's potential in an IoT context (§7), before describing how IFC integrated with middleware allows wide-scale application of this data-centric management approach, thus providing a solid foundation for building a legally-compliant IoT.

IFC research dates back to the 1970s [28] in the context of centralised military systems. Here, data was classified system-wide as *public, confidential, secret* or *top-secret*. Later, decentralised IFC was proposed [60], and has formed the basis of subsequent IFC research, including our work on *CamFlow* [68, 69]. In an OS implementation, IFC can be described as a data-centric, continuous (within the local system), Mandatory Access Control (MAC) mechanism.

Most IFC models, including our own, relate to two data properties: its *secrecy* and its *integrity*; respectively, where the data is allowed to flow to (per Bell and LaPadula [15]) and where it can flow from (per Biba [19]). These concerns are represented by associating with an entity A, two security labels $S(A)$ for secrecy and $I(A)$ for integrity; *active* (e.g. processes) and *passive* (e.g. data) entities are labelled. Our IFC model uses labels that comprise a set of tags, each tag representing a particular security concern (e.g. $S = \{medical\}$, $I = \{sanitised\}$). Tags are defined as required in order to represent policy, for example, relating to how personal medical data can flow. The **security context** of an entity is defined as the state of its two labels, S and I. The flow of data between entities composing the systems is only allowed towards equally or more constrained entities in order to guarantee for example, the proper usage of data.

These requirements are captured in the following constraints, which are applied on every data flow from an entity A to an entity B:

$$A \rightarrow B, \text{ iff } \{S(A) \subseteq S(B) \wedge I(B) \subseteq I(A)\}$$

Creation flows. If an entity *creates* an entity (active or passive), the created entity inherits the labels of its parents. In OS-level IFC enforcement, examples of entity creation include a process creating a file, and forking a child process.

Privileges for label change. In addition to their S and I labels, certain entities may have privileges to add and/or remove tags from these labels. An active entity may have four privilege tag sets in addition to its security context. These indicate its privilege to add and/or remove tags in the sets to/from its S and I labels. Though a created entity inherits the labels (security context) of its creator, privileges are not inherited and have to be passed explicitly.

Tag Ownership. In some IFC models [59], the concept of tag ownership is used to assign privileges to an entity. Privileges must be passed on with care, especially a privilege to remove a tag from a label (see below).

Security contexts. The security context of an entity is its pair of labels, S and I. A security context domain comprises entities with the same labels. The flow of data can therefore be within a security context domain or into a more constrained domain. Once data has flowed into a more constrained domain, further flows are confined to that domain or into increasingly constrained domains. For example, Fig. 3 shows that data tagged as s_1 can flow to an entity tagged with $S = \{s_1, s_2\}$ but then can only flow within the $S = \{s_1, s_2\}$ domain. Generally, building a system with increasing constraints can lead to situations of "label creep".

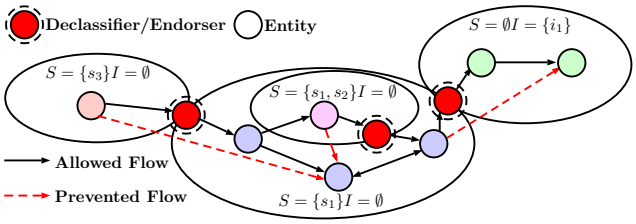

Figure 3: Declassification and endorsement

In practice, perhaps after a certain time has elapsed, secret data may need to be made publicly available, or when data has gone through an encryption or anonymisation process it is allowed to flow more freely. To achieve these things, an IFC system needs to support more complex flow policies.

Certain entities within an IFC system have privileges to modify their labels to transfer information across security contexts. An entity changing its security context is called a *declassifier* when it modifies secrecy constraints, and an *endorser* when it modifies integrity constraints.

Endorsers/declassifiers can be seen as trusted gateways between security context domains, where IFC constraints would otherwise prohibit a direct flow (see Fig. 3). Such gateways can help ensure that regulation is enforced, e.g., medical data might only flow to a research domain if it has gone through a declassifier that applies a specified anonymisation algorithm. As well as such *transformation* of the data, checks such as the time the data is authorised to be released might also be needed. These gateway processes will play a crucial role in enforcing policy at scale, as discussed in [66].

7 IFC's potential for the IoT

We now present a medical home-monitoring example, to show the potential of IFC in an IoT environment. For this example we assume policy is defined by the hospital, in terms of any access control credentials and IFC tags, and that middleware manages the components.

Fig. 4 shows a scenario where patients are discharged from hospital to home monitoring. Each patient has a dedicated Hospital Data Analyser to analyse and archive some of their data, which is sent in certain circumstances. If this component detects that a medical condition needs attention, hospital staff are alerted and the home sensors may be actuated to sample more frequently, as shown in Fig. 7. To ensure confidentiality, Ann's device has secrecy tags *medical* and *ann*. Only a component with these tags can receive data from Ann's device. The Analyser can only receive data in hospital-standard format from consenting patients, indicated by the integrity tags *hosp-dev* and *consent*. Ann's device is issued by the hospital and is tagged *hosp-dev*. Fig. 4 shows that data from Zeb cannot flow to Ann's Analyser, failing both the secrecy and integrity checks (both of which must be satisfied for a flow to occur).

Fig. 5 shows a component that sanitises non-standard input. It acts as an endorser, being privileged to input non-standard data tagged with e.g. $I = \{zeb\text{-}dev, consent\}$, convert the data to the required format then change its security context (a privileged action) so that Zeb's data can flow to

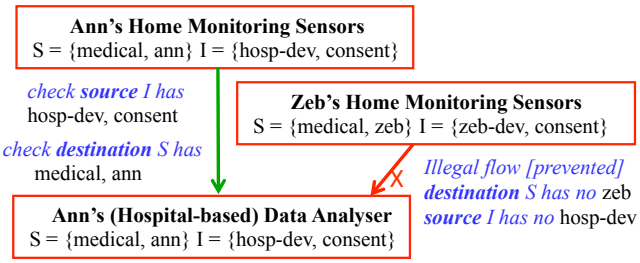

Figure 4: Home monitoring information flow

his Hospital Data Analyser. This process chain, from sensor through endorser to analyser is invoked dynamically whenever data should be transferred to the Analyser.

Note that even within this single application (home monitoring), access control alone cannot capture the fact that the sanitising component can receive data from Zeb's non-standard device when in one security context and can send data to the analyser only after changing its security context.

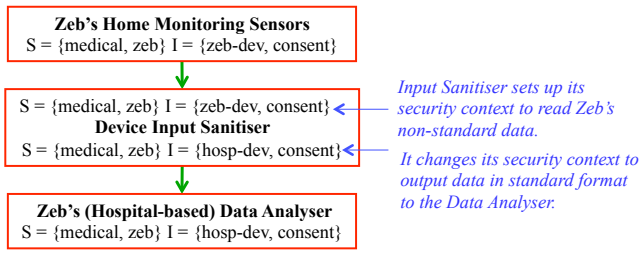

Figure 5: Endorsement in IFC

Fig. 6 illustrates *declassification*. It shows a component that generates statistics about the home monitoring initiative. Patients are assured that their personal data will not be discernible from the statistics; and regulation and policy dictate that the statistical use must entail anonymisation (according to some approved algorithm). We first see the statistics generator labelled to be able to read the personal data from all the patients. It carries out anonymisation and statistics generation then changes its security context to $S = \{medical, stats\}, I = \{anon\}$, before outputting its results to management (similarly labelled). Management cannot see patients' data directly.

Again, note that standard access controls alone cannot enforce the policy that only after the data is anonymised can it flow to management. The IFC enforcement logs will show that the statistics generator changed its security context before passing on the data. If a leak of personal data is claimed, the audit logs can be inspected to check for all flows relating to that data, and the statistics generating component's algorithm can be investigated.

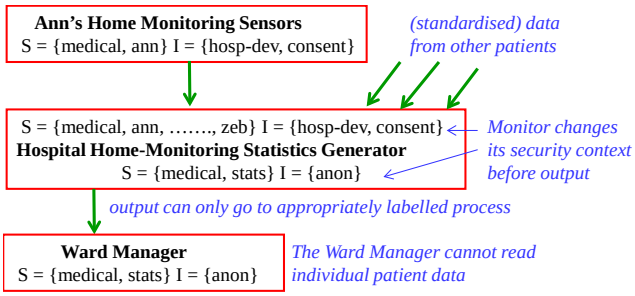

Figure 6: Declassification in IFC

A high-level overview of the entire home monitoring system is shown in Fig. 7. The patients' Data Analysers (Ann's is shown) monitor their conditions and if a medical emergency is detected, policy must come into force (red arrows) to alert emergency services and to change the monitoring parameters by an actuation command. Here, an application-aware policy engine triggers the middleware to set up the required new connections and set the security regime for the relevant components to ensure a compliant system. Managing IFC tags is discussed in §8 and as a challenge in §9.

Appropriately tagged components can be added, e.g. for medical research into different diseases. The example could be extended with patients needing to give consent for particular types of medical research, captured in the integrity tags of their data, and in the components handling the data; again an aspect of the challenges (in §9).

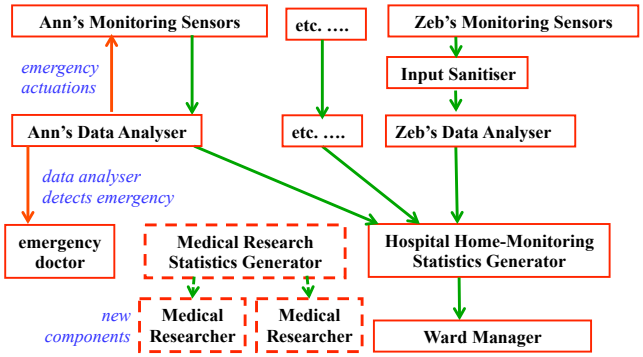

Figure 7: A home monitoring system

These examples motivate the use of IFC for protected data sharing in the IoT to augment conventional (principal cf. data-centric) access controls. Concerns 1, 2 and 3 of §3 are addressed directly, and IFC contributes to meeting Concerns 4, 5 and 6, discussed in §9.

8 Work towards the vision

We argue that the IoT-wide application of IFC will assist with issues of data management, and we have shown its conceptual fit by means of an example. However, the vision of enabling a policy-driven, and therefore more legally-compliant IoT must be grounded in practical realities. Towards this, we now describe our related systems experience of dynamically reconfigurable MW, through work on SBUS [82], and on IFC for cloud, though CamFlow [69].

8.1 Dynamically reconfigurable MW

The importance of MW with reconfiguration capabilities, towards enabling a more legally-compliant IoT, is that it allows system components to be changed and adapted when and where appropriate. This enables the system to accord with the general governance regime.

As described in §5.2, in a data management context we consider reconfiguration functionality as providing the means to: (a) change security regimes of components to define the bounds within which data flows take place; and (b) actively effect security operations by establishing or ceasing interactions, to directly initiate or prevent certain data flows.

Through our work on SBUS [82], we have long considered the practical aspects of reconfigurable MW. SBUS has a general AC regime to govern interactions. This policy, encapsulating attributes of principals and context, is enforced

at the granularity of message type,[2] and can be reconfigured.

Of particular relevance here is that SBUS not only supports system components reconfiguring their own state; but importantly, allows reconfiguration actions to be issued by *third parties*. That is, certain components can *instruct* others to undertake reconfigurations and actions.

These third-party instructions are executed as though the application had initiated them; though they occur independently from the application logic of the component being reconfigured. The reconfiguration commands are issued through the messaging system via control messages. This is shown in Fig. 8. Reconfiguration commands are subject to the same general AC regime, to ensure that reconfigurations are only actioned when received from *trusted* third parties.

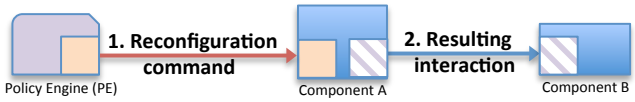

Figure 8: Third-party reconfiguration: a control message initiating a new interaction

The importance of this approach to reconfiguration is that it allows components to be externally managed. *This paves the way for policy to apply across systems.* That is, policy actions can entail the issuing of reconfiguration instructions to affect a range of components, in order to make the broader system behave according to high-level goals.

Such capabilities will be crucial for enabling IFC to apply at scale. Specifically, we anticipate reconfigurations will be the means to change components' security contexts, assign/remove privileges, etc. ((a), above); and also to enable transparent and dynamic system chain management ((b), above), for instance, to automatically include various declassifiers/endorsers and associated transformation operations to allow data to flow across IFC security context domains [66].

In practice, we envisage *policy engines*, entities that encapsulate a range of related policies, monitor environments and use the MW's remote-reconfiguration functionality to issue instructions to components, when/where necessary, to ensure system behaviour remains appropriate over time. For instance, in [81] we explored how policy engines in an assisted living environment could reconfigure a range of system components to enact an appropriate emergency response.

8.2 CamFlow: IFC for cloud

CamFlow [68, 69] realises the IFC model described in §6 to provide the continuous protection of data as it flows end-to-end through a PaaS cloud. This is achieved by enforcing IFC (1) at the OS kernel level, for entities co-hosted in the same OS instance, including for inter-process communication; and (2) at a message-passing level, for entities hosted by different OS instances (cross-machine). Therefore in CamFlow, IFC policy is enforced across cloud-hosted applications.

An underlying assumption is that the IFC implementation (and therefore the cloud-provider) is trusted. We argue that this is reasonable in a cloud context, because a cloud provider is generally more trustworthy, has a greater technical capability, and is more visible to regulators than tenants and their applications. Further, this means that cloud-hosted parties can collaborate without trusting each other, so long as they all trust the underlying IFC enforcement mechanism of the platform.

[2]Privileges, credentials and context are represented as X.509 certificates, managed and assigned using the standard mechanisms of the certificate infrastructure.

8.2.1 OS-level IFC enforcement

CamFlow provides a kernel level IFC-enforcement capability, to both enforce (control) and record data flows between processes and kernel objects (e.g. files, pipes, etc.).[3] This is implemented as a *Linux Security Module* [92] (LSM). LSMs use security hooks that are invoked on system calls to decide whether a call is allowed to proceed. LSMs associate with each kernel object a structure for storing security metadata comprising the object's security context and privileges (§6). Importantly, LSMs can be incorporated with limited overhead, leaving the rest of the kernel unaltered and system calls unchanged. This approach means IFC can be enforced without any application rework, intervention or even awareness. Further, all data flows can be tracked to enable audit, provenance and potentially demonstrate compliance with contracts and regulations. We have shown the LSM performance overhead to be minimal [68].

The LSM is simply the IFC enforcement mechanism. At a higher level, in decentralised IFC, labels and tags are defined for applications in order to reflect policy, and an application instance must be set up in an appropriate security context.

8.2.2 Cross-machine enforcement

The LSM can only enforce IFC within the context of the local OS. Unmediated external communication of labelled processes is prevented, since the context of security across the remote machine/network is unknown to the kernel.

Transfers across machines are therefore managed by a trusted substrate. This substrate represents an extension of our work in *SBUS*, where each communicating entity (application process) is associated with a messaging substrate process for external transfers. A substrate process is aware of the security context of the application process it serves, and enforces IFC in its dealings with the substrate processes of other applications. Fig. 9 shows the architecture.

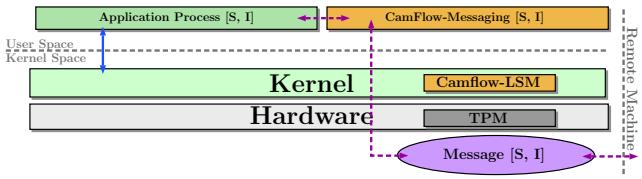

Figure 9: CamFlow architecture

Enforcement occurs on the establishment of communication (messaging) channels. A channel is only established if the policy allows, i.e. the tags of the components accord. Specifically, this involves augmenting the standard MW AC (principal and contextual policy) enforcement with a subsequent evaluation of IFC policy that encapsulates the data-centric constraints. This is monitored throughout the connection's lifetime, where an entity changing its security context triggers re-evaluation (enforcement). Note, the impacts of externally reconfiguring IFC policy as described in §8.1 have not been explored, remaining an open challenge (§9).

Message-specific policy. Though CamFlow enables enforcement without application intervention, there may be some circumstances where an application wishes to define its own high-level IFC policy. Messages are strongly typed,

[3]See www.camflow.org for the LSM IFC implementation.

consisting of a set of named and typed attributes, and certain message types, or attributes thereof, can be more sensitive than others; e.g. for a message type *person*, attribute *name* is likely more sensitive than *country*.

To achieve these more granular controls, additional tags can be defined that only exist at the messaging level, augmenting the OS-level security context. This is shown in Fig. 10, where tag C is message-layer specific. Enforcement may entail source quenching, in that messages/attribute values are not transferred if the tags of each party do not accord. This represents the integration and enforcement of IFC by systems operating at different technical levels. The ability for IFC policy to interoperate, and be enforced at different levels is crucial for enabling IFC to apply at scale.

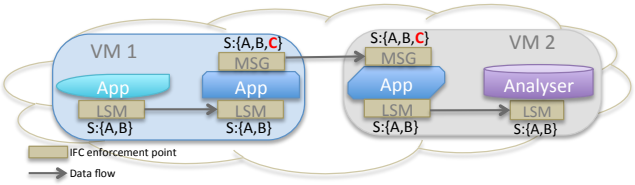

Figure 10: A messaging substrate (MSG) enforcing an additional app-specific policy (tag C)

8.3 Tracking data flow: provenance and audit

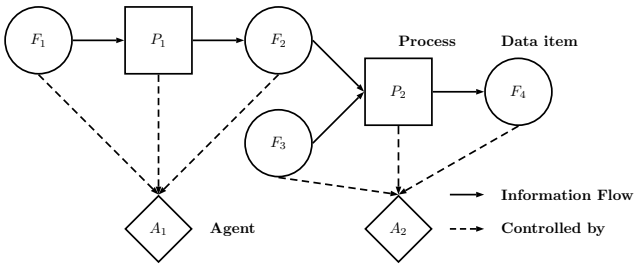

Figure 11: Fragment of an audit graph

A data provenance system assists in understanding data lifecycles: *how was it created? when? by whom? how was it manipulated?* [23]. Provenance systems generate audit graphs (a fragment is shown in Fig. 11), that represent the data items (F), transformation processes (P) and agents (A) (owners/managers) involved in generating and transferring certain data. These graphs are useful in forensic analysis, e.g. determining how a file was generated [51].

As both provenance and IFC concern the flow of information between entities, the logs generated during IFC enforcement are a natural source of provenance information. This was explored in [68], where we showed how a popular graph database (Neo4J[4]) and visualisation tool (Cytoscape[5]) can be used to analyse IFC audit data. Logs can be made more trustworthy by, for example, using hardware cryptographic support [6] as discussed in §4.

It follows that in addition to controlling data flows, IFC enforcement can produce provenance-like data that allows information transfers to be audited. These audits can be used to demonstrate system adherence to policy, and therefore compliance with various legal obligations.

[4]http://neo4j.com/
[5]http://js.cytoscape.org/

9 Moving forward

We have described the nature of IoT data sharing, considering its dynamic, *ad hoc* and federated nature, and the need for data exchanges to be managed in order to ensure those involved in the IoT comply with their legal obligations. Our big idea is that IFC—enabling data flow-centric control and audit—working with a middleware providing dynamic reconfigurable management functionality, offers real potential towards realising the vision of a legally-complaint IoT.

9.1 From clouds to the IoT

Our work on IFC for PaaS clouds shows IFC as a viable technology. This, however, was based on the assumption that the OS (cloud-provider) is trusted, possibly backed by hardware (see [67], §4)—justifiable given cloud is often an environment under a single domain of administrative control. Our initial work indicates that IFC-encoded policy can be both simple and expressive. We have also explored [86] how IFC might address enforcement of, and compliance with, the emerging legal/regulatory framework for cloud services [54].

This work has given us confidence that IFC has great potential for better enabling IoT applications to be legally-compliant, essential for realising the broader IoT vision.

9.2 IFC in meeting IoT data sharing concerns

Let us first reconsider the potential of IFC for addressing the concerns presented in §3:

Concern 1 "IoT involves highly sensitive data". Traditional access control can be used for principal-based, point-based, application-specific access checks. IFC tags are designed specifically to augment this for continuous access enforcement, including data secrecy and integrity aspects, as noted for the patient home monitoring example (Figs. 5, 6, 7), while offering potential for audit and provenance tracking.

Concern 2 "IoT involves actuation". Hardware-based attestation and traditional access controls can be augmented by IFC's integrity tags, designed to ensure the 'quality' (e.g. authority, accuracy, geo-location, etc.) of the actuation command. This allows not only the issuer and context immediately surrounding the command to be considered, but also the chain of flows related to the command, including the data (e.g. event patterns) that led to the actuation.

Concern 3 "Persistence of policy through composed things". Assuming policy permits some composition of components, IFC can enforce all flows between them, and offer data protection beyond the lifetime of the composition (and the components involved). This assumes that tags are allocated to authorised components before flows can take place.

Concern 4 "Data quality and provenance". IFC integrity tags are designed to ensure and enforce data quality properties, e.g. as shown in Fig. 5 to indicate data has been converted to a standard format. In general, integrity tags are assigned after data has been validated. This might be after hardware certification of certain properties (as discussed in §4). For example, [44] uses hardware cryptographic support and RFID tracking to certify the physical (GPS) location of machines. A similar mechanism could be used to guarantee sensor accuracy or other physical properties.

For data provenance, IFC checks are carried out on every attempted flow. This facilitates the creation of logs recording all attempted and permitted flows. Such information provides the means to demonstrate that user policies have been enforced and regulations have been complied with. It

also provides a basis for investigating attacks, the effects of which may be confined by the IFC control regime.

Concern 5 "Data aggregation, inference and analytics". The problem of unintended and unforeseen inference is unsolved and involves keeping a step ahead of ever-more sophisticated algorithms. IFC can contribute by logging where data travels, and when combined with other data [86]. Regulation is involved in controlling analytics and IFC can assist in making evident what data has been used, when and by whom.

Concern 6 "Customisation and context". Declassifier/endorsement processes allow data to flow between security context domains, where data flows would otherwise be prohibited. This allows constraints on data to change over time, as appropriate to the situation. For example, after a certain period of time, governmental data previously considered secret should become public. This involves changing constraints, through a managed, explicit process of an entity modifying its security context to declassify the data.

9.3 Research challenges

The discussion of §9.2 considers the potential of IFC for meeting specific IoT concerns. The fundamental challenge in realising the big idea is making IFC apply at scale, given:

1. The heterogenous nature of the chains of IoT components, which exist across federated domains of administration; and
2. The IoT environments are long-lived, yet highly dynamic.

Therefore we believe MW with reconfiguration capabilities has a clear role to play. To reiterate, MW can naturally operate across system and administrative boundaries, managing interactions, and therefore is highly appropriate for integrating IFC functionality. By enabling external reconfiguration, the security policy of components can be changed and adapted to ensure that the system remains compliant over time, and given changes in circumstances. This is crucial for realising a legally-compliant IoT.

However, there are a number of significant challenges that must first be overcome to realise the grand vision; given these concern dynamism and cross-system aspects, we believe the middleware community is well placed to contribute. We now summarise some key areas for research.

Challenge 1: Global policy. For security policy to apply at scale, throughout the IoT, there is a need for a global policy representation, including tag and privilege descriptions, and a universal means for reconfiguring components.

Current IFC implementations have privileged application managers that can create application-specific IFC tags [69]. In general, privileged processes can arbitrarily create tags that may be application-/system-specific, e.g. data in an EU cloud may be tagged as such, to comply with regulation that personal data must not leave the EU [39, 86]. However, data can only flow after negotiation and subsequent propagation of tags, so this must be managed. Though some policies may only be relevant within a particular scope (e.g. an application), others may be globally applicable, and interactions may occur with entities never before encountered.

In short, complexity is exacerbated when moving beyond the single administrative domain of a cloud service. Therefore, there must be the potential for (certain) security policy to *potentially* apply to any 'thing', through changes in context, wherever it lies, in any IoT processing chain. With tags, one way forward may be approaches akin to DNS and/or based on PKI, though overheads will be a consideration.

There is also the need for reconfiguration operations to be standardised, to allow components' security policy to be changed where necessary, to align with broader, system-wide requirements. Though we argue that the MW should provide the reconfiguration functionality, the precise definition and nature of the operations must be globally understood.

Also relevant is that policy can apply at different levels of abstraction; e.g. in our own work (§8), translation is necessary between the kernel's tag representation and that of the messaging substrate that deals with other machines. This requires consideration as more technologies are involved.

Challenge 2: Defining policy. Our discussion of the viability of our big idea has focussed on the mechanism for enforcement. However, for the vision to be realised, there is a clear need for suitable, intuitive means for IFC tags, privileges and reconfiguration policy to be expressed, so that obligations can be captured and adhered to. Work concerning policy authoring interfaces and templates can be relevant, as is work on ontologies that relate to policy semantics. It is also important to recognise that the tags may themselves be sensitive e.g. where a tag implies a particular medical condition. Therefore, the visibility of policy specifications may also need to be controlled.

Challenge 3: Managing reconfiguration. We argue that reconfiguration mechanisms are required given that systems are long lived and requirements will change over time.

Policy uses reconfiguration functionality to respond to changing circumstances, where actions are taken on patterns of events, e.g. detected by complex-event methods or machine learning. Therefore *representing and managing context* is an ongoing challenge, particularly given the IoT's scale.

Beyond context, of crucial importance is how these reconfigurations are properly managed—not least as 'things' can be reused, repurposed, mobile, shared, and usage can vary. Investigation is required on how the continuous data-centric control mechanisms resolve with more general AC, and how reconfigurations can be used to set up security contexts for new applications and service-composition lifecycles, and manage these for dynamic application. In particular, IFC imposes a stringent regime, where deliberate behaviours (endorsement/declassification) must be undertaken for data to flow beyond defined boundaries. How the external reconfiguration of IFC policy (label assignment, privileges) issued by others should be managed to avoid components reaching an inconsistent state, through the updated policy regime, is an open challenge. Much work is required on the feedback loops between the policy, reconfiguration/enforcement mechanisms and components.

Challenge 4: Authority and conflict. Given the IoT is federated by nature, one issue concerns managing who is able to define and maintain (reconfigure) policy. Some 'things' are owned by individuals, e.g. wearables; some are shared, e.g. the occupants of a home may all have the right to control certain 'things' therein; and some devices have delegated ownership, e.g., a health service may loan devices to patients, or a company to employees. There may also be *ad hoc* situations, in which some authority is given temporarily, e.g. only while physically in a particular location.

Related is that federation means that policy will conflict, particularly as many of these considerations will be contextual and policy can change. We have previously considered

dynamic policy conflict [83], including policy prioritisation and override, but only where the conflict is within a single administrative domain. Work is certainly required on policy conflict resolution, e.g. standardisation, authoring interfaces and/or mechanisms for runtime negotiation and resolution.

Challenge 5: Trusted enforcement. Clearly a policy enforcement infrastructure must be trustworthy, particularly if the goal is to assist with compliance. In §8.2 we mentioned that a cloud provider is likely to be trusted and therefore it may be reasonable that the OS kernel is part of a trusted computing base for enforcing IFC, especially if hardware-backed [67]. This can motivate sharing between applications hosted in the cloud, as they need only trust the cloud provider's enforcement mechanism.

While this assists some forms of cloud-mediated IoT services, enabling trust in the wider IoT environment is challenging. Federation means a variety of parties will own, control and manage system components, and the physical nature of 'things' raises issues of system integrity (tampering). Further, there may be dynamically-formed interactions with components never before seen, managed by parties relatively unknown. Hardware-based trust approaches, e.g. those described in §4, show promise by improving the level of trust and decreasing the size of the trusted software base [5,17,61].

Resource constraints are another consideration: some devices may have a limited ability to store and enforce policy. Of course, gateway components could be used to mediate data flows. However, substantial work is required on what aspects of policy management and enforcement can be delegated, offloaded, distributed and federated, to meet resource constraints, while ensuring adequate levels of trust.

Challenge 6: Audit and provenance for compliance. Our goal of a legally-compliant IoT is based on the ability to provide evidence that data management responsibilities are being met. We described how IFC facilitates data provenance, including when data is amalgamated with other data, providing evidence for audit and compliance.

Though we have explored audit in a single administrative domain (§8.3) managing provenance in IoT is difficult, due to its federated and dynamic nature. Questions include: What should be recorded, and when? How to deal with possible audit 'gaps', where components are no longer accessible, intermittently connected or mobile? Are there any special considerations where interactions are one-off and *ad hoc*? How does provenance work, given the resource constraints of 'things'? When can logs safely be pruned? Can logs be offloaded to others for distributed audit, and how should this be managed? Can audit be reconciled with legal obligations for non-disclosure, e.g. actions by law enforcement?

The above challenges, though significant, are within scope of the middleware research community's expertise. Our discussion concerns the IoT, but work in such areas will assist in building more legally-compliant systems in general, which is of increasing importance given ongoing legal developments.

10 Related Work

Surveys have shown that issues concerning security, privacy and data management are the key research challenges for the IoT [8,34,89], and compliance is clearly related [85]. We now summarise areas of related research.

10.1 Current IoT security research

The IoT raises security challenges at all technical levels [75]. There is much work on establishing secure communication channels between 'things' [31,37]; standards for this purpose are summarised in [46]. Others have designed access control schemes specifically for IoT [25, 63], and some research focuses on lightweight authentication mechanisms [49, 65, 74].

These approaches, however, do not consider protecting data beyond a particular interaction, i.e. do not provide continuous, system-wide guarantees. They also tend to focus on adapting existing techniques (§4) to resource-constrained devices, rather than devising new approaches towards general IoT requirements. Such work is orthogonal to, but complements our vision, e.g. helping to integrate low-end devices.

It is generally recognised that high-level management concerns, which at a systems level relate particularly to issues of policy, reconfiguration and context management, remain open issues given IoT's massive scale [34, 89].

10.2 Middleware for the IoT

Middleware has a clear role in the IoT [13] to assist with issues of interoperability, service composition, heterogeneity, layers of abstraction, security/privacy, context-awareness, resource management, QoS, etc. In this *big ideas* paper, we have established additional requirements for IoT, relating to continuous system-wide policy enforcement to enable legal compliance. These concerns align with the general directions of emerging middleware, towards managing future complex and dependable systems [43]. To place our idea in context, we briefly outline the most relevant areas.

Service composition and adaptive middleware. *Service composition* entails the combination of services (or system components) to realise particular functionality [42]. This typically involves taking an application level task (request) and mapping it to a number of services [45], managing resource allocation and task distribution/ordering. Service composition is important for IoT, where coordinating and orchestrating 'things' helps meet functional goals. As such, there is work, e.g. [33, 91] on supporting service composition at scale, accounting for high levels of heterogeneity.

Middleware allowing dynamic configuration and customisation is known as *adaptive* [76] or *reflective* [48]. These approaches expose the current system configuration and state, enabling reconfigurations to fulfil resource or QoS requirements [1]. Reconfigurations may be initiated by applications themselves, or by managers of lower-level (e.g. network)/ shared infrastructure. Service composition is closely related, as reconfigurations enable orchestrations to be effected.

This paper argues that reconfigurations are necessary for IoT, to help manage the dynamic runtime environment. Both adaptive and service composition aspects are useful in helping to reconfigure and orchestrate components, to realise particular functionality. However, there is also the need for higher level controls through policy, not only to serve application requirements, but broader policy requirements across applications, systems and services.

Specifically, we see the need for complementary control mechanisms that apply continuously. That is, data management policy must operate beyond the scope of a particular service composition to: (1) regulate the data flows within any composition, and in, out and/or between them, system-wide; and (2) account for any resulting data that may exist

well beyond the lifetime of the composition. Secondly, there must be means to reconfigure, and adapt the policy itself.

Policy-based systems. There is some work on higher-level policy enforcement [88]. Policy models, however, typically involve imposing an architectural or environmental structure. For example, *self-managed cells* [27] were proposed to bind a group of 'things' in order to manage the interactions with other cells, and in [56] agents are grouped with trusted controllers. Other constraints may be imposed, e.g. forcing a particular form of interaction [53, 83, 93].

We have argued the case for a policy-driven IoT, however IoT requires systems to be open and flexible. Thus, any policy-based infrastructure must avoid imposing restrictive modelling, design or interaction constraints, so that any required structuring is dictated by users. That is, it is important that data flow policy is enforced *regardless of any system structure*, for the entire lifecycle of the data. Further, having a policy model that is deliberately flat enables decentralised and flexible, granular policy that can apply *directly*, rather than be limited by layers of abstraction.

Sticky policies have been proposed to achieve end-to-end control over data [21, 71], where data is encrypted along with the policy to be applied to that data. To obtain the decryption key from a *Trusted Authority*, a party must agree to enforce the policy. Sticky policies aim at a high level of abstraction, where policy definition and interpretation are comparatively heavyweight. Further, the approach is trust-based with no audit of compliance; there are no means to ensure the proper usage of data once decrypted.

Note that translating law into a machine understandable form is an active area of research of the computational law community [16, 50]. Though such work typically does not consider systems-level enforcement, it could play a useful role in assisting policy definition.

Representing context. IoT is dynamic and data-driven, therefore context is a key consideration [72]. Policy is inherently contextual, defined to be enforced in particular circumstances. Therefore, a richer representation of state allows for more granular and expressive policy.

A well-structured context model enables analysis and reasoning over policy behaviours, useful for determining operational semantics, potential policy or contextual errors/omissions. There are various approaches enabling complex state representations [12, 18]. Ontological approaches show particular promise [20, 36], by allowing context, tags, privileges, etc. to be defined, based on semantics. However, given the scale of IoT, managing context is an ongoing concern; indeed, agreeing a common vocabulary/coding scheme, even for subareas of a heterogeneous IoT is challenging [64].

10.3 IFC in the IoT

A survey of IFC implementations is given in [9]. The majority of approaches concern enforcement within a local machine; very few consider IFC across machines [24,79,95], and these tend to be limited in scope and/or impose a certain form of system structuring (see [84] for more discussion). Our own work (§8) is alone in proposing IFC for cloud computing. To our knowledge, IFC has not been considered for the widely-distributed, highly dynamic environment of the IoT, which as this paper describes, entails a number of significant research challenges.

10.4 Audit and provenance for the IoT

It is often acknowledged that audit will be important for the IoT, but there is little work in the area. Policy enforcement, coupled with audit for a widely distributed context, is yet to be addressed.

Provenance tools associated with OSs or hypervisors [52, 58, 73] aim at logging data at a lower systems-level, useful for system administration. Cloudopsy [94], focusses on the visualisation of information flows for end users, for investigation of privacy issues, but without addressing enforcement. SPADE [32] provides graph-based models of the flow of scientific data in grid computing, following the Open Provenance Model.[6] The EU Compose project proposes a Provenance Data Model for the IoT [14]; this reinforces the need for confidentiality and integrity enforcement mechanisms.

11 Conclusion

System design is increasingly affected by the development and application of law and regulation, particularly with regard to data management. The future IoT will be a highly distributed, dynamic, data-driven environment, yet compliance with data management obligations must be demonstrated during system operation.

We have argued that a new approach must be taken to enable the *big idea of a legally-compliant IoT*. This is where policy, aligning with legal obligations, is able to drive and manage system behaviour, and where the infrastructure provides evidence to assist in demonstrating compliance.

Given that the IoT is driven by the exchange of data, we feel that middleware with reconfiguration capabilities, integrated with IFC, represents a promising way forward. This is by providing the means to support managed and adaptable data sharing policy, the continuous monitoring and management of data flows, and logging for compliance and audit. We have shown the viability of IFC as a systems technology, in the context of a trusted cloud. However, the scale and dynamic nature of the IoT means major research challenges remain in a number of areas, to enable such technology to realise a legally-compliant IoT.

Though such research is, without doubt, relevant for maximising the IoT's potential, we believe work in the areas described would also identify new directions for investigation, leading to a new, interdisciplinary general research topic: *"legally-compliant distributed systems"*.

Acknowledgements

Work based in the Computer Laboratory was supported by the UK Engineering and Physical Sciences Research Council grant EP/K011510 CloudSafetyNet: End-to-End Application Security in the Cloud, and Microsoft through the Microsoft Cloud Computing Research Centre.

12 References

[1] Combining heterogeneous service technologies for building an Internet of Things middleware. *Computer Communications*, 35(4):405 – 417, 2012.

[2] Overview of the Internet of Things. Technical Report Y.2060, ITU Telecommunication Standardization Sector, June 2012.

[6] http://openprovenance.org

[3] Software Guard Extensions Programming Reference. (Intel, Technical Report 329298-001US), 2013.

[4] Internet of Things (Preliminary Report 2014). (Technical Report, ISO/IEC JTC 1), 2015.

[5] N. Aaraj, A. Raghunathan, and N. K. Jha. Analysis and Design of a Hardware/Software Trusted Platform Module for Embedded Systems. *Transactions on Embedded Computing Systems (TECS)*, 8(1):8, 2008.

[6] R. Accorsi. BBox: A distributed secure log architecture. In *Public Key Infrastructures, Services and Applications*, pages 109–124. Springer, 2011.

[7] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy re-Encryption Schemes with Applications to Secure Distributed Storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[8] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[9] J. Bacon, D. Eyers, T. Pasquier, J. Singh, I. Papagiannis, and P. Pietzuch. Information Flow Control for Secure Cloud Computing. *Transactions on Network and System Management SI Cloud Service Management*, 11(1):76–89, 2014.

[10] J. Bacon, K. Moody, and W. Yao. A Model of OASIS Role-based Access Control and its Support for Active Security. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):492–540, 2002.

[11] S. Bajikar. Trusted Platform Module (TPM) based Security on Notebook PCs-White Paper. *Mobile Platforms Group, Intel Corporation*, pages 1–20, 2002.

[12] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, June 2007.

[13] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta. Role of middleware for Internet of Things: A study. *International Journal of Computer Science and Engineering Survey*, 2:94–105, Aug 2011.

[14] S. Bauer and D. Schreckling. Data Provenance in the Internet of Things. In *EU Project COMPOSE, Conference 2013*.

[15] D. E. Bell and L. J. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report M74-244, The MITRE Corp., Bedford MA, 1973.

[16] T. Bench-Capon, M. Araszkiewicz, K. Ashley, et al. A History of AI and Law in 50 Papers: 25 Years of the International Conference on AI and Law. *Artif. Intell. Law*, 20(3):215–319, Sept. 2012.

[17] S. Berger, K. Goldman, D. Pendarakis, D. Safford, E. Valdez, and M. Zohar. Scalable Attestation: A Step Toward Secure and Trusted Clouds. In *International Conference on Cloud Engineering (IC2E)*. IEEE, 2015.

[18] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180, 2010.

[19] K. J. Biba. Integrity Considerations for Secure Computer Systems. Technical Report ESD-TR 76-372, MITRE Corp., 1977.

[20] G. S. Blair, A. Bennaceur, N. Georgantas, P. Grace, V. Issarny, V. Nundloll, and M. Paolucci. The role of ontologies in emergent middleware: Supporting interoperability in complex distributed systems. In *ACM/IFIP/USENIX Middleware 2011, Springer LNCS 7049*, pages 410–430, 2011.

[21] D. W. Chadwick and S. F. Lievens. Enforcing sticky security policies throughout a distributed application. In *Workshop on Middleware Security*, pages 1–6. ACM, 2008.

[22] D. W. Chadwick, A. Otenko, and E. Ball. Role-based Access Control with X. 509 Attribute Certificates. *Internet Computing, IEEE*, 7(2):62–69, 2003.

[23] A. Chapman, M. D. Allen, and B. T. Blaustein. It's About the Data: Provenance as a Tool for Assessing Data Fitness. In *Workshop on the Theory and Practice of Provenance*. USENIX, 2012.

[24] W. Cheng, D. R. K. Ports, D. Schultz, V. Popic, A. Blankstein, J. Cowling, D. Curtis, L. Shrira, and B. Liskov. Abstractions for Usable Information Flow Control in Aeolus. In *USENIX Annual Technical Conference*, Boston, 2012.

[25] A. Cherkaoui, L. Bossuet, L. Seitz, G. Selander, and R. Borgaonkar. New Paradigms for Access Control in Constrained Environments. In *9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–4. IEEE, 2014.

[26] J. Crowcroft, A. Madhavapeddy, M. Schwarzkopf, T. Hong, and R. Mortier. Unclouded Vision. In *Distributed Computing and Networking*, pages 29–40. Springer, 2011.

[27] P. De Leusse, P. Periorellis, T. Dimitrakos, and S. K. Nair. Self Managed Security Cell, a Security Model for the Internet of Things and Services. In *1st International Conference on Advances in Future Internet*, pages 47–52. IEEE, 2009.

[28] D. E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.

[29] C. Dwork. Differential privacy. In *Automata, Languages and Programming*, pages 1–12. Springer, 2006.

[30] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. (IETF Technical Report), 2002.

[31] O. Garcia-Morchon, S. Kumar, R. Struik, S. Keoh, and R. Hummen. Security Considerations in the IP-based Internet of Things. IETF, 2013.

[32] A. Gehani and D. Tariq. SPADE: Support for Provenance Auditing in Distributed Environments. In *ACM/IFIP/USENIX Middleware*, pages 101–120. Springer, 2012.

[33] P. Grace, Y.-D. Bromberg, L. Réveillère, and G. Blair. Overstar: An open approach to end-to-end middleware services in systems of systems. In *ACM/IFIP/USENIX Middleware*, pages 229–248. Springer, 2012.

[34] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[35] W. Hartzog and E. Selinger. The Internet of Heirlooms and Disposable Things. *North Carolina Journal of Law & Technology*, 581, June 2016.

[36] S. Hasan and E. Curry. Thingsonomy: Tackling Variety in Internet of Things Events. *Internet Computing*, 19(2):10–18, Mar 2015.

[37] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle. Security Challenges in the IP-based Internet of Things. *Wireless Personal Communications*, 61(3):527–542, 2011.

[38] W. K. Hon, C. Millard, and J. Singh. Twenty Legal Considerations for Clouds of Things. (Queen Mary University of London, School of Law, Technical Report 216/2016), 2016.

[39] W. K. Hon, C. Millard, J. Singh, I. Walden, and J. Crowcroft. Policy, legal and regulatory implications of a Europe-only cloud. *International Journal of Law and Information Technology*, 2016.

[40] D. Hrestak and S. Picek. Homomorphic Encryption in the Cloud. In *Proc. 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1400–1404. IEEE, 2014.

[41] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle. Towards Viable Certificate-based Authentication for the Internet of Things. In *2nd Workshop on Hot Topics in Wireless Network Security and Privacy*, pages 37–42. ACM, 2013.

[42] N. Ibrahim and F. Le Mouël. A survey on service composition middleware in pervasive environments. *International Journal of Computer Science Issues*, 1:1–12, Aug 2009.

[43] V. Issarny and G. Blair. Guest editorial: Special issue on the future of middleware (FOME'11). *Journal of Internet Services and Applications*, (1):1–4, May.

[44] K. R. Jayaram, D. Safford, U. Sharma, V. Naik, D. Pendarakis, and S. Tao. Trustworthy Geographically Fenced Hybrid Clouds. In *ACM/IFIP/USENIX Middleware*. ACM, 2014.

[45] S. Kalasapur, M. Kumar, and B. Shirazi. Dynamic service composition in pervasive computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(7):907–918, 2007.

[46] S. L. Keoh, S. Kumar, and H. Tschofenig. Securing the Internet of Things: A Standardization Perspective. *Internet of Things Journal*, 1(3):265–275, 2014.

[47] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang. Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence. In *Dependable Systems & Networks (DSN'09)*, pages 115–124. IEEE, 2009.

[48] F. Kon, F. Costa, G. Blair, and R. H. Campbell. The case for reflective middleware. *Communications of the ACM*, 45(6):33–38, Jun 2002.

[49] J.-Y. Lee, W.-C. Lin, and Y.-H. Huang. A Lightweight Authentication Protocol for Internet of Things. In *International Symposium on Next-Generation Electronics (ISNE)*, pages 1–2. IEEE, 2014.

[50] N. Love and M. Genesereth. Computational law. In *10th International Conference on Artificial Intelligence and Law*, pages 205–209. ACM, 2005.

[51] R. Lu, X. Lin, X. Liang, and X. S. Shen. Secure Provenance: the Essential of Bread and Butter of Data Forensics in Cloud Computing. In *Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 282–292. ACM, 2010.

[52] P. Macko, M. Chiarini, and M. Seltzer. Collecting Provenance via the Xen Hypervisor. In *TaPP*. USENIX, 2011.

[53] N. Matthys, C. Huygens, D. Hughes, J. Ueyama, S. Michiels, and W. Joosen. Policy-driven tailoring of sensor networks. In *Springer, Sensor Systems and Software, S-CUBE'10*, pages 20–35, 2010.

[54] C. J. Millard, editor. *Cloud Computing Law*. Oxford University Press, 2013.

[55] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma. A gap analysis of Internet-of-Things platforms. *Computer Communications*, 2016.

[56] N. H. Minsky and V. Ungureanu. Law-governed interaction. *ACM Transactions on Software Engineering Methodologies*, 9(3):273–305, 2000.

[57] G. Mulligan. The 6LoWPAN architecture. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 78–82. ACM, 2007.

[58] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer. Provenance-aware storage systems. In *USENIX Annual Technical Conference*, pages 43–56, 2006.

[59] A. C. Myers. JFlow: Practical Mostly-static Information Flow Control. In *26th SIGPLAN SIGACT POPL'99*, pages 228–241. ACM, 1999.

[60] A. C. Myers and B. Liskov. A Decentralized Model for Information Flow Control. In *Symposium on Operating Systems Principles (SOSP)*, pages 129–142. ACM, 1997.

[61] M. Nauman, S. Khan, X. Zhang, and J.-P. Seifert. Beyond Kernel-level Integrity Measurement: Enabling Remote Attestation for the Android Platform. In *Trust and Trustworthy Computing*, pages 1–15. Springer, 2010.

[62] Nuffield Council on Bioethics. *The collection, linking and use of data in biomedical research and health care: ethical issues*. 2014.

[63] S. W. Oh and H. S. Kim. Decentralized Access Permission Control Using Resource-oriented Architecture for the Web of Things. In *Conference on Advanced Communication Technology (ICACT)*, pages 749–753. IEEE, 2014.

[64] M. Paolucci and B. Souville. Data interoperability in the future of middleware. *Journal of Internet Services and Applications*, 3(1):127–131, May 2012.

[65] N. Park, M. Kim, and H.-C. Bang. Symmetric Key-Based Authentication and the Session Key Agreement Scheme in IoT Environment. In *Computer Science and its Applications*, pages 379–384. Springer, 2015.

[66] T. Pasquier, J. Bacon, J. Singh, and D. Eyers. Data-Centric Access Control for Cloud Computing. In *Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 2016.

[67] T. Pasquier, J. Singh, and J. Bacon. Clouds of Things need Information Flow Control with Hardware Roots

of Trust. In *International Conference on Cloud Computing Technology and Science (CloudCom'15)*. IEEE, 2015.

[68] T. Pasquier, J. Singh, J. Bacon, and D. Eyers. Information Flow Audit for PaaS clouds. In *International Conference on Cloud Engineering (IC2E)*, pages 42–51. IEEE, 2016.

[69] T. Pasquier, J. Singh, D. Eyers, and J. Bacon. CamFlow: Managed Data-Sharing for Cloud Services. *IEEE Transactions on Cloud Computing*, 2015.

[70] S. Pearson. Trusted Computing Platforms, the Next Security Solution. *HP Labs*, 2002.

[71] S. Pearson and M. Casassa-Mont. Sticky Policies: An Approach for Managing Privacy across Multiple Parties. *Computer*, 44, July 2011.

[72] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the Internet of Things: A survey. *Communications Surveys Tutorials, IEEE*, 16(1):414–454, First 2014.

[73] D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler. Hi-Fi: Collecting High-Fidelity whole-system provenance. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 259–268. ACM, 2012.

[74] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov, and M. Ylianttila. Two-phase Authentication Protocol for Wireless Sensor Networks in Distributed IoT Applications. In *14th Int. Conf. on Wireless Communications and Networking (WCNC)*, pages 2770–2775. IEEE, 2014.

[75] R. Roman, P. Najera, and J. Lopez. Securing the Internet of Things. *Computer*, 44(9):51–58, 2011.

[76] S. M. Sadjadi and P. K. McKinley. A survey of adaptive middleware. *Michigan State University Report MSU-CSE-03-35*, 2003.

[77] N. Santos, H. Raj, S. Saroiu, and W. A. Using ARM TrustZone to Build a Trusted Language Runtime for Mobile Applications. In *Proc. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 67–80. ACM, 2014.

[78] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-based Cloudlets in Mobile Computing. *Pervasive Computing, IEEE*, 8(4):14–23, 2009.

[79] L. Sfaxi, T. Abdellatif, R. Robbana, and Y. Lakhnech. Information Flow Control of Component-based Distributed Systems. *Concurrency and Computation: Practice and Experience*, 25(2):161–179, 2013.

[80] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). *IETF Standards Track*, 2014.

[81] J. Singh and J. Bacon. On Middleware for Emerging Health Services. *Journal of Internet Services and Applications*, 5(6):1–34, 2014.

[82] J. Singh, D. Eyers, and J. Bacon. Policy Enforcement within Emerging Distributed, Event-Based Systems. In *Distributed Event-Based Systems (DEBS'14)*, pages 246–255. ACM, 2014.

[83] J. Singh, D. M. Eyers, and J. Bacon. Disclosure Control in Multi-Domain Publish/Subscribe Systems. In *Distributed Event-Based Systems (DEBS'11)*, pages 159–170. ACM, 2011.

[84] J. Singh, T. Pasquier, J. Bacon, and D. Eyers. Integrating Middleware and Information Flow Control. In *International Conference on Cloud Engineering (IC2E)*, pages 54–59. IEEE, 2015.

[85] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers. Twenty security considerations for cloud-supported Internet of Things. *IEEE IoT Journal*, 3(3):269–284, June 2016.

[86] J. Singh, J. Powles, T. Pasquier, and J. Bacon. Data flow management and compliance in cloud computing. *IEEE Cloud Computing Magazine, Special Issue on Legal Clouds*, 2(4):24–32, July 2015.

[87] J. Singh, L. Vargas, J. Bacon, and K. Moody. Policy-based Information Sharing in Publish/Subscribe Middleware. In *Policy*. IEEE, 2008.

[88] M. Sloman. Policy Driven Management For Distributed Systems. *Journal of Network and Systems Management*, 2:333–360, 1994.

[89] J. Stankovic. Research Directions for the Internet of Things. *Internet of Things Journal*, 1(1):3–9, 2014.

[90] S. Subashini and V. Kavitha. A Survey on Security Issues in Service Delivery Models of Cloud Computing. *Journal of Network and Computer Applications*, 34(1):1 – 11, 2011.

[91] T. Teixeira, S. Hachem, V. Issarny, and N. Georgantas. Service oriented middleware for the Internet of Things: A perspective. In *ServiceWave'11*, pages 220–229, 2011.

[92] C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman. Linux Security Modules: General security support for the Linux kernel. In *Foundations of Intrusion Tolerant Systems*, pages 213–213. IEEE, 2003.

[93] A. Wun and H.-A. Jacobsen. A policy management framework for content-based publish/subscribe. In *ACM/IFIP/USENIX Middleware*, pages 368–388. Springer, 2007.

[94] A. Zavou, V. Pappas, V. P. Kemerlis, M. Polychronakis, G. Portokalidis, and A. D. Keromytis. Cloudopsy: An autopsy of data flows in the cloud. In *Human Aspects of Information Security, Privacy, and Trust*, pages 366–375. Springer, 2013.

[95] N. Zeldovich, S. Boyd-Wickizer, and D. Mazières. Securing Distributed Systems with Information Flow Control. In *5th Symposium on Networked System Design and Implementation (NSDI 08)*, pages 293–308. USENIX, 2008.